Version: 6.x

Moving between screens

In the previous section, "Hello React Navigation", we defined a stack navigator with two routes (`Home` and `Details`), but we didn't learn how to let a user navigate from `Home` to `Details` (although we did learn how to change the *initial* route in our code, but forcing our users to clone our repository and change the route in our code in order to see another screen is arguably among the worst user experiences one could imagine).

If this was a web browser, we'd be able to write something like this:

```
<a href="details.html">Go to Details</a>
```

Another way to write this would be:

```
<a
  onClick={() => {
    window.location.href = 'details.html';
  }}
>
  Go to Details
</a>
```

We'll do something similar to the latter, but rather than using a `window.location` global, we'll use the `navigation` prop that is passed down to our screen components.

Navigating to a new screen

```
import * as React from 'react';
import { Button, View, Text } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details')}
      />
    </View>
  );
}

// ... other code from the previous section
```

Try this example on Snack

Let's break this down:

- `navigation` - the `navigation` prop is passed in to every **screen component** (definition) in the native stack navigator (more about this later in "The navigation prop in depth").
- `navigate('Details')` - we call the `navigate` function (on the `navigation` prop — naming is hard!) with the name of the route that we'd like to move the user to.

> If we call `navigation.navigate` with a route name that we haven't defined in a navigator, it'll print an error in development builds and nothing will happen in production builds. Said another way, we can only navigate to routes that have been defined on our navigator — we cannot navigate to an arbitrary component.

So we now have a stack with two routes: 1) the `Home` route 2) the `Details` route. What would happen if we navigated to the `Details` route again, from the `Details` screen?

Navigate to a route multiple times

```
function DetailsScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Details Screen</Text>
      <Button
        title="Go to Details... again"
        onPress={() => navigation.navigate('Details')}
      />
    </View>
  );
}
```

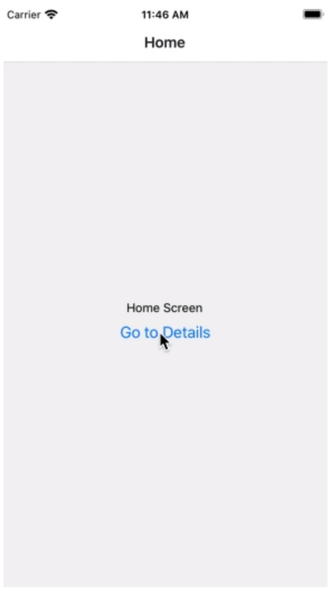
Try this example on Snack

If you run this code, you'll notice that when you tap "Go to Details... again" that it doesn't do anything! This is because we are already on the Details route. The `navigate` function roughly means "go to this screen", and if you are already on that screen then it makes sense that it would do nothing.

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.

```
<Button
  title="Go to Details... again"
  onPress={() => navigation.push('Details')}
/>
```

Try this example on Snack



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.

Going back

The header provided by the native stack navigator will automatically include a back button when it is possible to go back from the active screen (if there is only one screen in the navigation stack, there is nothing that you can go back to, and so there is no back button).

Sometimes you'll want to be able to programmatically trigger this behavior, and for that you can use `navigation.goBack();`.

```
function DetailsScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Details Screen</Text>
      <Button
        title="Go to Details... again"
        onPress={() => navigation.push('Details')}
      />
      <Button title="Go to Home" onPress={() => navigation.navigate('Home')} />
      <Button title="Go back" onPress={() => navigation.goBack()} />
    </View>
  );
}
```

Try this example on Snack

On Android, React Navigation hooks in to the hardware back button and fires the `goBack()` function for you when the user presses it, so it behaves as the user would expect.

Another common requirement is to be able to go back *multiple* screens -- for example, if you are several screens deep in a stack and want to dismiss all of them to go back to the first screen. In this case, we know that we want to go back to `Home` so we can use `navigate('Home')` (not `push`! try that out and see the difference). Another alternative would be `navigation.popToTop()`, which goes back to the first screen in the stack.

```
function DetailsScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Details Screen</Text>
      <Button
        title="Go to Details... again"
        onPress={() => navigation.push('Details')}
      />
      <Button title="Go to Home" onPress={() => navigation.navigate('Home')} />
      <Button title="Go back" onPress={() => navigation.goBack()} />
      <Button
        title="Go back to first screen in stack"
        onPress={() => navigation.popToTop()}
      />
    </View>
  );
}
```

Try this example on Snack

Summary

- `navigation.navigate('RouteName')` pushes a new route to the native stack navigator if it's not already in the stack, otherwise it jumps to that screen.
- We can call `navigation.push('RouteName')` as many times as we like and it will continue pushing routes.
- The header bar will automatically show a back button, but you can programmatically go back by calling `navigation.goBack()`. On Android, the hardware back button just works as expected.
- You can go back to an existing screen in the stack with `navigation.navigate('RouteName')`, and you can go back to the first screen in the stack with `navigation.popToTop()`.
- The `navigation` prop is available to all screen components (components defined as screens in route configuration and rendered by React Navigation as a route).

Edit this page