Navigation and Routing in Flutter

Learning Outcomes

After completing this lesson you should be able to

- Explain what a Route means
- Explain different ways of Navigating between screens
- Explain how to pass arguments from one screen to another screen
- Explain different ways for extracting arguments
- Explain the difference between Navigator 1.0 and Navigator 2.0

Overview

In Flutter, `screens` and `pages` are called `routes`

In Android, a `route` is equivalent to an `Activity`

In Flutter, a `route` is just a `widget`

Navigating between two routes

Create two routes (pages/screens)

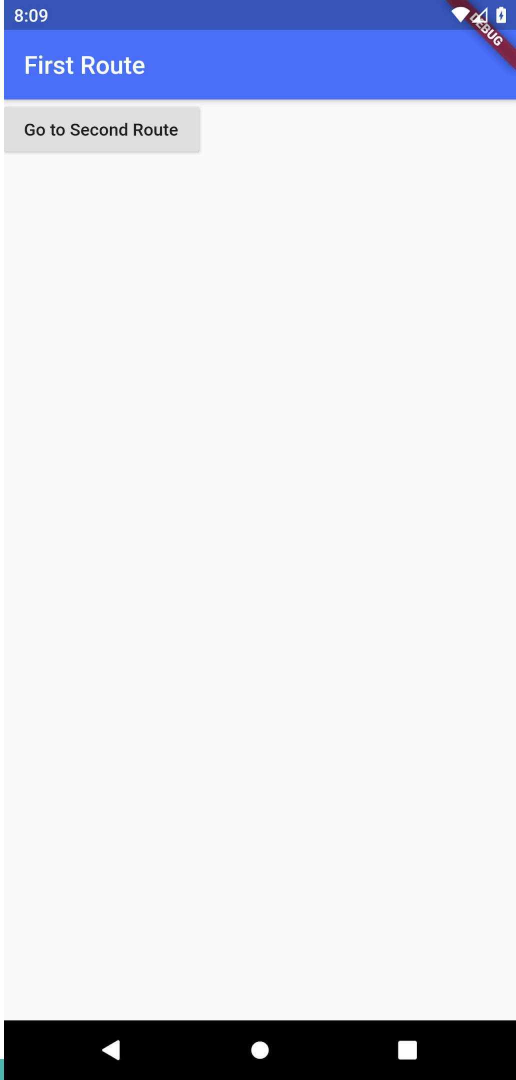
Navigate to the second route using `Navigator.push()`

> Navigate to a new screen by **creating a new route** and **pushing** it to the `Navigator`

Return to the first route using `Navigator.pop()`

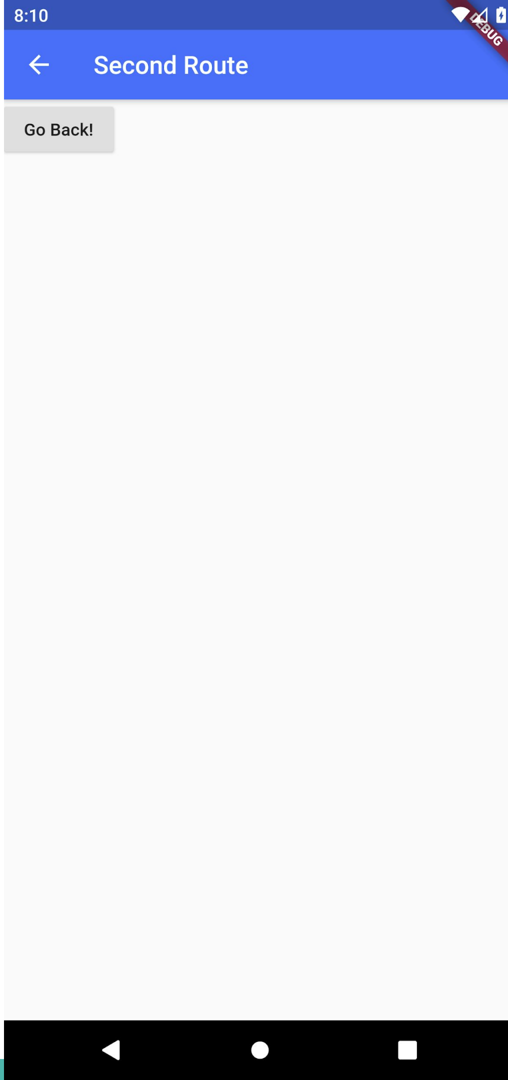
First Route/Page/Screen

```
class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext ctx) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Route'),
      ),
      Body: ElevatedButton(
          child: Text('Go to Second Route'),
          onPressed: () {
            // Navigate to second route when tapped.
          },
       ),
    );
  }
}
```



Second Route/Page/Screen

```
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext ctx) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: ElevatedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
          },
          child: Text('Go back!'),
      ),
    );
  }
```



Navigate to the second route

```
// Within the `FirstRoute` widget
onPressed: () {
  Navigator.push(
    ctx,
    MaterialPageRoute(builder: (ctx) => SecondRoute()),
  );
}
```

Return to the first route

```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(ctx);
}
```

Main function

```
import 'package:flutter/material.dart';

void main() {
 runApp(
   MaterialApp(
     home: FirstRoute(),
   ),
 );
}
```

Problem

If you need to navigate to the same screen in many parts of your app, this approach can result in code duplication

Navigate with named routes

To work with named routes, use the **`Navigator.pushNamed()`** function

- Create two screens
- Define the routes
- Navigate to the second screen using **`Navigator.pushNamed()`**
- Return to the first screen using **`Navigator.pop()`**

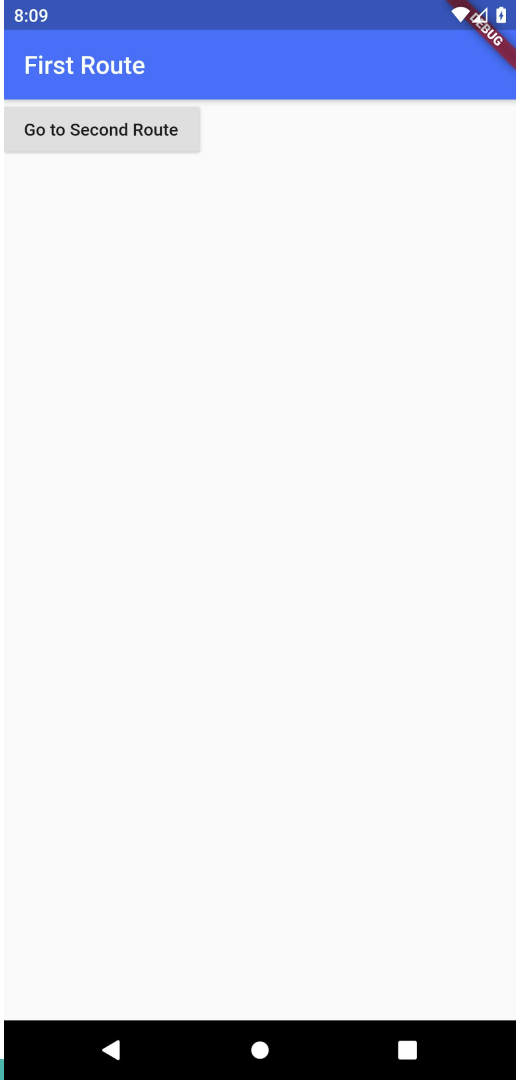
Create two screens

Let us use the same Routes we used before

They are shown in the next two slides

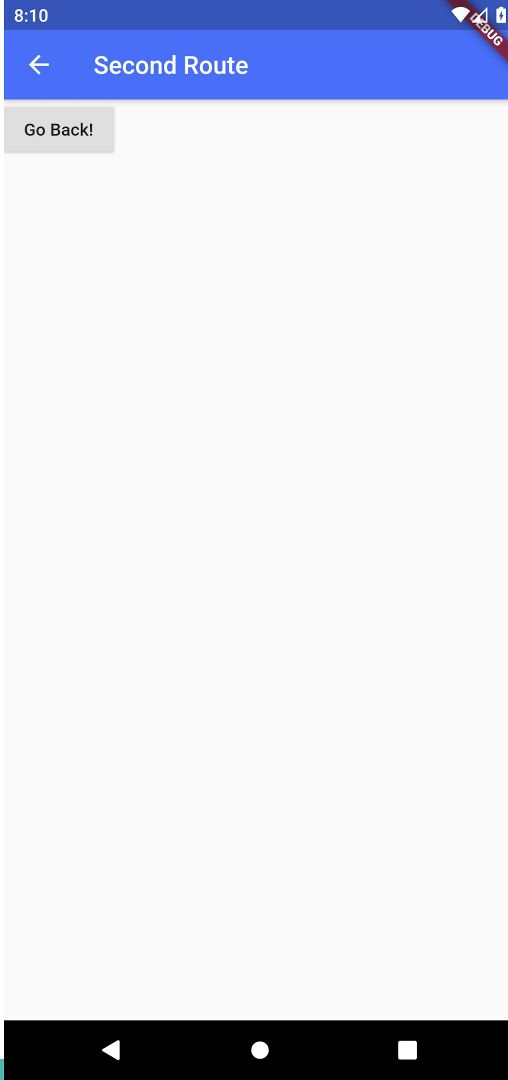
First Route/Page/Screen

```
class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext ctx) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Route'),
      ),
      Body: ElevatedButton(
          child: Text('Go to Second Route'),
          onPressed: () {
            // Navigate to second route when tapped.
          },
       ),
    );
  }
}
```



Second Route/Page/Screen

```
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext ctx) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: ElevatedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
          },
          child: Text('Go back!'),
      ),
    );
  }
```



Define the routes

```dart
MaterialApp(
  // Start the app with the "/" named route
  initialRoute: '/',
  routes: {
    // When navigating to the "/" route,
    // build the FirstRoute widget
    '/': (ctx) => FirstRoute(),
    // When navigating to the "/second" route,
    // build the SecondRoute widget
    '/second': (ctx) => SecondRoute(),
  },
);
```

Navigate to the second screen

```
// Within the `FirstRoute` widget
onPressed: () {
  // Navigate to the second screen using a named route.
  Navigator.pushNamed(ctx, '/second');
}
```

Return to the first screen

```
// Within the SecondScreen widget
onPressed: () {
  // Navigate back to the first screen
  // by popping the current route off the stack.
  Navigator.pop(ctx);
}
```

Pass arguments to a named route

You can use the `arguments` parameter of the `Navigator.pushNamed()` method to pass arguments to a named route

How to pass arguments to a named route

Define the arguments you need to pass

Pass the Argument

Extracts the arguments

Define the arguments you need to pass

Example, pass two pieces of data:

The **title** of the screen and a **message**

```
// You can pass any object to the arguments parameter
class RouteArguments {
  final String title;
  final String message;
  RouteArguments(this.title, this.message);
}
```

Pass the argument

Provide the arguments to the route via the **arguments** property of the `Navigator.pushNamed()` method of the `FirstRoute` widget

Check the next slide to see an example

```
class FirstRoute extends StatelessWidget {
  static const routeName = '/';
  @override
  Widget build(BuildContext ctx) {
    return Scaffold(
    // ...
      body: Center(
        child: ElevatedButton(
          child: Text('Go to Second Route'),
          onPressed: () {
            Navigator.pushNamed(
              ctx,
              SecondRoute.routeName,
              arguments: RouteArguments('Title Text','Message Text',),
            );
          },
        ),
      ),
    );
  }
}
```

Extract arguments

You can extract the arguments either using the `ModalRoute.of()` method or an `onGenerateRoute()` function provided to the `MaterialApp`

Here is what we want to do

> The `SecondRoute` widget will extract and use the argument passed to it by the `FirstRoute` widget

Extracts the arguments

```
class SecondRoute extends StatelessWidget {
  static const routeName = '/second';
  @override
  Widget build(BuildContext ctx) {
    final RouteArguments args = ModalRoute.of(ctx).settings.arguments;
    return Scaffold(
      appBar: AppBar(
        title: Text(args.title),
      ),
      body: Center(
        Text(args.message),
        child: ElevatedButton(
          // ...
        ),
      ),
    );
  }
```

Extracting arguments using `onGenerateRoute`

You can extract the arguments inside an `onGenerateRoute()` function and then pass them to a widget

Check the example shown in the next slide

You need to modify `SecondRoute` widget so that it accepts the arguments (**`title`** and **`message`**) as constructor argument

```
MaterialApp(

  onGenerateRoute: (settings) {
    if (settings.name == SecondRoute.routeName) {
      final RouteArguments args = settings.arguments;

        return MaterialPageRoute(
        builder: (context) {
          return SecondRoute(
            title: args.title,
            message: args.message,
          );
        },
      );
    }
  },
);
```

Return data from a route

You can use the `Navigator.pop()` method to return a data back to the previous screen

Let us say you want to display some text on a `SnackBar` on the `FirstRoute` after reading the text from the `SecondRoute`

The next two slides show how to achieve this

Return data from a route

Modify the **`Navigator.pop()`** method of the **`SecondRoute`** as shown below

```
    // ...

    RaisedButton(
      child: Text('Go Back!'),
      onPressed: () {
        Navigator.pop(ctx, 'Data from Second Route');
      },
    ),

  // ...
```

Return data from a route

Add the following property to **`FirstRoute`** widget class

```
class FirstRoute extends StatelessWidget {
 static const routeName = '/';

 final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();
    // ...
}
```

Return data from a route

Add the **`_scaffoldKey`** to the key property of the Scaffold widget of **`FirstRoute`** widget class

```
class FirstRoute extends StatelessWidget {
 // ...
 @override
 Widget build(BuildContext ctx) {
   return Scaffold(
     key: _scaffoldKey,
       // …
   )
}
```

Return data from a route

Modify the onPressed method of the RaisedButton inside `FirstRoute` as shown in the next slide so that it can fetch the data sent from `SecondRoute` and display it using `SnackBar`

```
class FirstRoute extends StatelessWidget {
        // ...
         RaisedButton(
           child: Text('Go to Second Route'),
           onPressed: () async {
             final String result =
                 await Navigator.push(ctx, MaterialPageRoute(builder: (ctx)
{
               return SecondRoute();
             }));
             final snackBar = SnackBar(content: Text(result));
             _scaffoldKey.currentState. showSnackBar(snackBar);
           },
    // ...
}
```

Send data to a new screen

Explore the example shown at the following link

`https://flutter.dev/docs/cookbook/navigation/passing-data`

Navigator 1.0 vs Navigator 2.0

Exercise:

Read about the new Navigator 2.0 following the link below

https://flutter.dev/go/navigator-with-router

And explain the difference between `Navigator 1.0` (all the previous slides are about `Navigator 1.0`) and `Navigator 2.0`

You need to be able to explain about the limitations of `Navigator 1.0` and how `Navigator 2.0` addresses those limitations