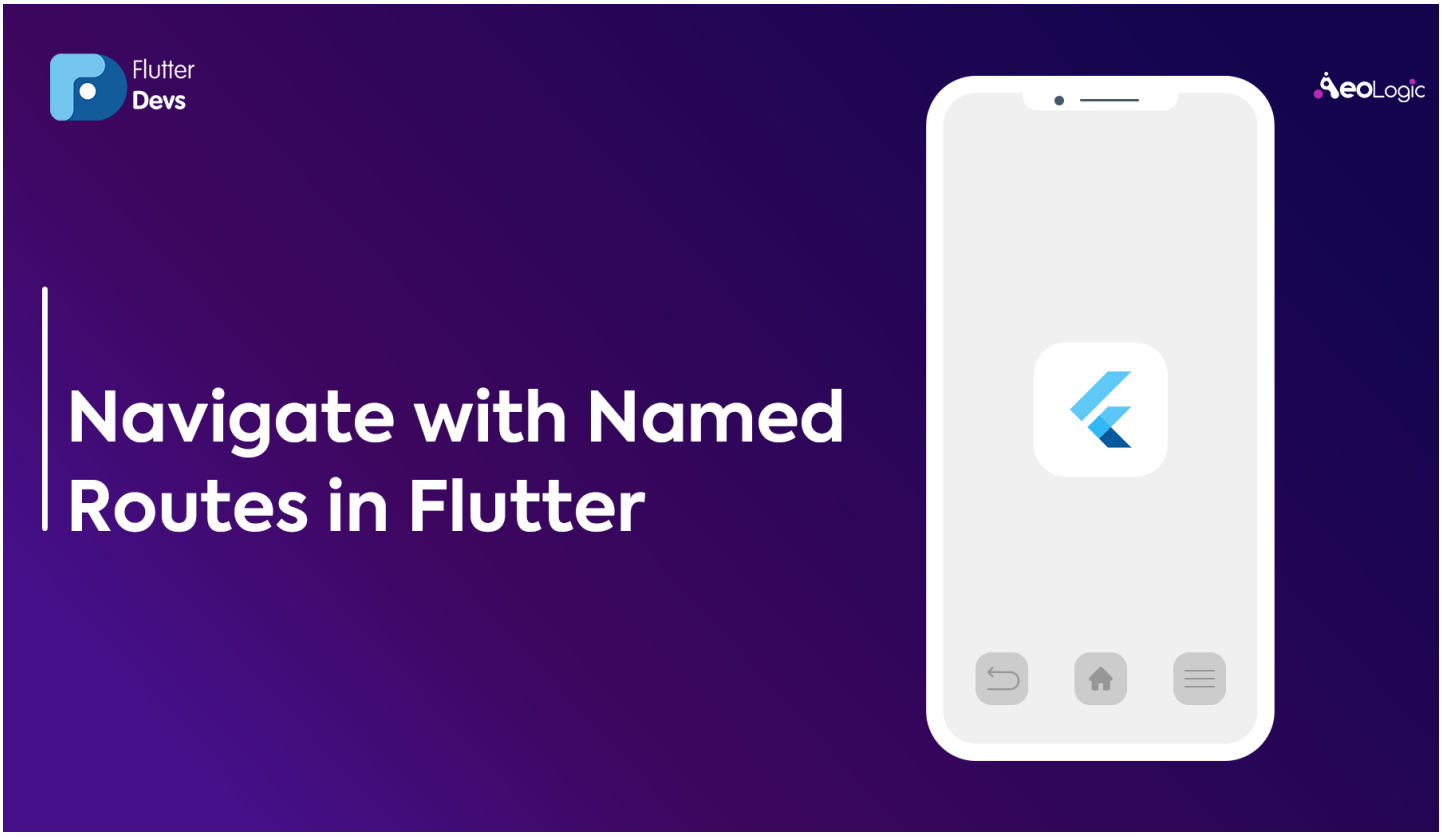
# Navigate with Named Routes in Flutter

Let's understand the idea of how to navigate to the different screens using the naming routes concept

Suraj Gupta Follow
Dec 11, 2020 · 4 min read




## Introduction :

Named Routes is the simplest way to navigate to the different screens using naming concepts. When a user needs multiple screens in an app depending on its need then we have to navigate from one screen to another and also return back many times back on the previous screen then it becomes a very hectic task, there we use a naming concept so as to remember screens with its name provided by the user and then we can directly access that route or page directly through Navigator.pushNamed() method.

## Table of content :



## Create two Routes

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


class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
     appBar: AppBar(
       title: Text('Home'),
     ),
     body: Center(
       child: RaisedButton(
         onPressed: (){
      // Navigate to next page ...
         },
         child: Text('Go to next'),
         ),
     ),
    );
  }
}
```

```
class Second extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
     appBar: AppBar(
       title: Text('Second Screen'),
     ),
     body: Center(
       child: RaisedButton(
         onPressed: (){
      // Navigate to previous ...
         },
         child: Text('Go back'),
         ),
     ),
    );
  }
}
```

Here above the code, we have created two routes or screens.

## Defining the routes with names :

```
MaterialApp(
      initialRoute: 'home',
      routes: {
        'home' : (context)⇒ Home(),
        'second': (context)⇒ Second()
      },
    );
```

**Note:** We cannot use **home** property and **initialRoute** property both inside this MaterialApp as they both contradict each other.

## Navigate from one route to another :

We will navigate from one screen to another using Navigator.pushNamed() method.

```
onPressed: (){
      // Navigate to second page ...
      Navigator.pushNamed(context, 'second');
         },
```

## Using static modifier to call naming routes :

If we use the routes method to navigate from one page to another with a naming concept then we have to remember the string that we have provided to its route. When there are multiple screens in an application then it becomes tough to remember each string. So we create one string datatype of it with the static modifier so that we can assign it by directly calling it through its class.

```
class Home extends StatelessWidget {

  static String id = 'home';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
     appBar: AppBar(
       title: Text('Home'),
     ),
     body: Center(
```

```
        child: RaisedButton(
          onPressed: (){

          },
          child: Text('Go to next'),
          ),
      ),
    );
  }
}

class Second extends StatelessWidget {

   static String id = 'second';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
     appBar: AppBar(
       title: Text('Second Screen'),
     ),
     body: Center(
       child: RaisedButton(
         onPressed: (){

         },
         child: Text('Go back'),
         ),
     ),
    );
  }
}
```

As you can see above we have used a static modifier with string datatypte to both our screen and assign it with the string variable.

***Now we have to see how we will call these class in our routes::***

```
MaterialApp(
      initialRoute: Home.id,
      routes: {
        Home.id : (context)⇒ Home(),
        Second.id: (context)⇒ Second()
      },
    );
```

As you can see now easily remember the routes by directly remembering its class name as shown above we have used `Home.id` and `Second.id` in place of using naming string routes and remembering each string name.

We can directly access it easily and can navigate from one page to another with the same name as provided above.

```
onPressed: (){
      // Navigate to next page ...
      Navigator.pushNamed(context, Second.id);
          },
```

> **For more info visit down the link:**

### Navigate with named routes

In the Navigate to a new screen and back recipe, you learned how to navigate to a new screen by creating a new route...

flutter.dev

> **Read my other blog :**

### Wrap class In Flutter

Let's understand how to use the Wrap widget in our basic UI☝

medium.com

Thanks for reading this article ♥

Clap 👏 If this article helps you.

Feel free to post any queries or corrections you think are required…✓

**FlutterDevs** team of Flutter developers to build high-quality and functionally-rich apps. Hire a flutter developer for your cross-platform Flutter mobile app project on an hourly or full-time basis as per your requirement! You can connect with us on Facebook, GitHub, Twitter, and LinkedIn for any flutter related queries.

We **welcome feedback** and hope that you share what you're working on using #**FlutterDevs**. We truly enjoy seeing how you use Flutter to build beautiful, interactive web experiences!.