Routing in Flutter

What are Routes?

In Flutter routes are basically instances of CupertinoPageRoute and MaterialPageRoute.

CupertinoPageRoute is for IOS apps

MaterialPageRoute is for Android Apps.

Managing Routes:

Navigator Class manages routes.

It works on the Stack Principle to place the pages/routes on top of each other.

To go to the new screen you need to push the route to the Stack while To go to previous screen you need to pop the route from the stack.

Methods for Defining Routes:

1- **Named Routes:**
   It provides centralized structural approach by naming each route.
   Example of Code Snippet using this method

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

void main(){

  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: '/',
      routes:{
        '/':(context)=>FirstPage(),
        '/ second':(context)=>SecondPage(),
      })
  );
```

```
}

class FirstPage extends StatelessWidget{

  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar:AppBar(
        title:const Text('FirstPage',),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body:Container(
        child:Center(
          child: ElevatedButton(onPressed: (){
            Navigator.pushNamed(context,'/ second');
          }, child: const Text('Move to Second Page')),)
      )
    );
  }
}

class SecondPage extends StatelessWidget{
  @override

  Widget build(BuildContext context){
    return Scaffold(
      appBar:AppBar(
        title:const Text('Second Page'),
        centerTitle: true,
        backgroundColor: Colors.pinkAccent,
      ),
      body:Container(
        child:Center(
          child:ElevatedButton(
            onPressed:(){
              Navigator.pop(context);
            },
            child:const Text('Go to First Page'),
          )
        )
      )
    );
  }
}
```

2- **Direct Route:**
This method passes the route directly to the CupertinoPageRoute or MaterialPageRoute. This method pushes a new route onto the stack with context referring to the location of the current screen.
Example of this is same as above but with only few changes

```
void main(){

  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      home:FirstPage(),
    ),
  );
}

class FirstPage extends StatelessWidget{

  @override
  Widget build(BuildContext context){
    return Scaffold(
      appBar:AppBar(
        title:const Text('FirstPage',),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body:Container(
        child:Center(
          child: ElevatedButton(onPressed: (){
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context)=>SecondPage()),
            );
          }, child: const Text('Move to Second Page')),)
      )
    );
  }
}

class SecondPage extends StatelessWidget{
  @override

  Widget build(BuildContext context){
    return Scaffold(
      appBar:AppBar(
        title:const Text('Second Page'),
        centerTitle: true,
        backgroundColor: Colors.pinkAccent,
      ),
      body:Container(
        child:Center(
          child:ElevatedButton(
            onPressed:(){
              Navigator.pop(context);
            },
            child:const Text('Go to First Page'),
          )
        )
      )
    );
  }
}
```

Passing Data Between Different Screen :

For this purpose, **Navigator.push** and **Navigator.pushNamed** are used

Example Code

```
Navigator.push(
                    context,
                    MaterialPageRoute(builder:(context)=>
                    SecondPage(data: "Data From First Page"))
                 );
```

In the SecondPage() there is a constructor that accepts this data

```
SecondPage({required this.data});
```