Flutter Navigation and Routing

In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing.
Flutter provides a basic routing class MaterialPageRoute and three methods Navigator.push(), Navigator.pushReplacement(), Navigator.pop() that shows how to navigate between Pages.

• Navigator.push :-

The Navigator.push() method is used to navigate/switch to a new page/screen. Here, the push() method adds a page on the stack and then manage it by using the Navigator.

```
onPressed: () {
 Navigator.push( context,  MaterialPageRoute( builder: (
context) => SecondRoute()  ),
 );  }
```

• Navigator.pop:-

we need to use Navigator.pop() method to close the second route and return to the first route. The pop() method allows us to remove the current route from the stack, which is managed by the Navigator.

```
 onPressed: () {
  Navigator.pop(context);  }
```

Flutter Basic UI elements cheat Sheet

• Navigator.pushReplacement :-

If we don’t want to let a user go back to the previous page, we need to remove the previous page from the stack. We don’t actually need to remove it. pushReplacement function does it for us.

```
Navigator.pushReplacement(
context, MaterialPageRoute(builder: (context) => SecondPage() ), );
```

• Navigator.pushNamed :-

when you have multiple routes in your app then you should use Navigator.pushNamed(). To identify and differentiate between multiple routes, we can give a name to the routes. This makes easy to navigate between different routes.

```
Navigator.pushNamed(context, 'routeName');
```

```
MaterialApp(
// Start the app with the "homeScreen" named route.
initialRoute: 'homeScreen',
routes: {
// When navigating to the "homeScreen" route.
'homeScreen': (context) => HomeScreen(),
// When navigating to the "secondScreen" route,.
'secondScreen': (context) => SecondScreen(), }, );
```

Flutter Splash Screen

A splash screen is an initial screen that gets displayed right when the user launches the app, before the main page loads. It may not look like much because it's only shown for a short time. But splash screens can really pack a punch as they're the first impression of the app.

```
class SplashScreen extends StatefulWidget {
 @override
 _SplashScreenState createState() => _SplashScreenState();  }

class _SplashScreenState extends State<SplashScreen> {

 @override
 void initState() {
 super.initState();
 Future.delayed( Duration(seconds: 2),
 () { Navigator.pushReplacementNamed(
context, '/home'); }, ); }

 @override
 Widget build(BuildContext context) {
 return Scaffold( body: Center( child: Column(
 mainAxisAlignment: MainAxisAlignment.center,
 children: <Widget>[ FlutterLogo( size: 100.0, ),
 SizedBox(height: 16.0),
 Text('Your Nursery App'), ], ), ), ); } }
```

Flutter Basic UI elements cheat Sheet

Flutter Bundle Passing

"Bundle passing" typically refers to passing data or parameters between different parts of an application or between different screens in a mobile app.
In the context of Flutter (or any mobile app development framework), "bundle passing" might not be the most commonly used term. However, the concept of passing data between different parts of an application or between screens is fundamental.

```
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(   appBar: AppBar(
title: Text('Flutter Demo'),   ),
      body: Center(   child: ElevatedButton(
        onPressed: () {   Navigator.push(   context,
MaterialPageRoute(
            builder: (context) =>
                    SecondPage(data: 'Hello from first page'),
          ),  );   },
        child: Text('Go to Second Page'),  ), ), ); } }

class SecondPage extends StatelessWidget {
  final String data;
  SecondPage({required this.data});

  @override
  Widget build(BuildContext context) {
    return Scaffold(    appBar: AppBar(
title: Text('Second Page'),  ),
      body: Center(   child: Text(data),   ),  );   } }
```