Flutter Navigation: Managing Screens & Routes

Subtitle: Learn how to navigate between screens efficiently in Flutter apps.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Navigation is essential for **moving between different screens** in an app. Flutter provides **Navigator, routes, and named routes** to handle screen transitions. This guide teaches the basics and best practices.

Step 1: Simple Navigation with Navigator.push

```
ElevatedButton(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );
  },
  child: Text('Go to Second Screen'),
)
```

Exercise: Create a button that navigates to a new screen displaying some text.

Step 2: Navigate Back

```
ElevatedButton(
  onPressed: () {
    Navigator.pop(context);
  },
  child: Text('Go Back'),
)
```

Exercise: Add a back button on the second screen to return to the first screen.

Step 3: Passing Data Between Screens

```
// Sending
```

```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => SecondScreen(data: 'Hello from First Screen'),
  ),
);

// Receiving
class SecondScreen extends StatelessWidget {
  final String data;
  const SecondScreen({required this.data, Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(child: Text(data)),
    );
  }
}
```

Exercise: Pass a username or any string from one screen to another.

Step 4: Named Routes

```
void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => FirstScreen(),
      '/second': (context) => SecondScreen(),
    },
  ));
}
```

Exercise: Navigate to `/second` using `Navigator.pushNamed(context, '/second')`.

Step 5: Passing Data with Named Routes

```
Navigator.pushNamed(
  context,
  '/second',
  arguments: 'Hello via Named Route',
```

```
);

class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final data = ModalRoute.of(context)!.settings.arguments as String;
    return Scaffold(body: Center(child: Text(data)));
  }
}
```

Exercise: Pass and display a custom message using named routes.

Step 6: Mini Project

Build a **Multi-Screen App**:

- Home screen with buttons to navigate to 2–3 other screens
- Pass data between screens
- Use both `Navigator.push` and named routes
- Add back buttons on all screens

Key Takeaways

- `Navigator.push` and `Navigator.pop` handle basic navigation
- Named routes make **large apps easier to manage**
- Passing data allows **dynamic content on new screens**
- Proper navigation is essential for **user-friendly apps**

Skipping proper navigation makes your app **static and difficult to use**, which frustrates users.

Visit **haas.dev** for more Flutter guides, tutorials, and complete project resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
