Flutter Navigation & Routing: Master Screens & Routes

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

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

Introduction

Every app has multiple screens. Proper navigation is crucial to ensure a **smooth user experience**. Flutter provides **Navigator, routes, and named routes** to manage screen transitions. This guide teaches the basics and best practices.

Step 1: Basic Navigation with Navigator.push

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => SecondScreen()),
);
```

Exercise: Navigate from a Home screen to a Details screen on button press.

Step 2: Returning Data from a Screen

```
final result = await Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => InputScreen()),
);

print("Returned value: $result");
```

Exercise: Create a screen where user enters their name and return it to Home screen.

Step 3: Using Navigator.pop

```
ElevatedButton(
  onPressed: () {
    Navigator.pop(context, "Hello Home!");
  },
```

```
  child: Text("Go Back"),
)
```

Exercise: Pass data back from second screen to first.

Step 4: Named Routes

1. Define routes in `MaterialApp`:

```
MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => HomeScreen(),
    '/details': (context) => DetailsScreen(),
  },
)
```

2. Navigate using route names:

```
Navigator.pushNamed(context, '/details');
```

Exercise: Convert your app to use named routes.

Step 5: Passing Arguments with Named Routes

```
Navigator.pushNamed(
  context,
  '/details',
  arguments: 'Hafsa',
);
```

Retrieve arguments in DetailsScreen:

```
final args = ModalRoute.of(context)!.settings.arguments as String;
```

Exercise: Pass username to Details screen and display it.

Step 6: Pop Until / Remove Screens

- Pop back to a specific screen:

```
Navigator.popUntil(context, ModalRoute.withName('/'));
```

- Replace screen completely (no back button):

```
Navigator.pushReplacementNamed(context, '/details');
```

Exercise: Create a login flow that replaces login screen after successful login.

Step 7: Mini Project

Build a **Multi-Screen App**:

- Home Screen with navigation buttons
- Details Screen that accepts arguments
- Return data to Home Screen
- Use named routes and pushReplacement for login flow

Key Takeaways

- Navigator.push / pop = basic navigation
- Named routes improve scalability
- Passing arguments enables dynamic screens
- Proper route management is crucial for **smooth UX**
- Master navigation to avoid messy screen flows

Ignoring proper navigation makes your app **hard to use and unprofessional**, even if your features work.

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

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