

# Biomedical Wearable Technologies for Healthcare and Wellbeing

## Navigation

---

A.Y. 2021-2022

Giacomo Cappon

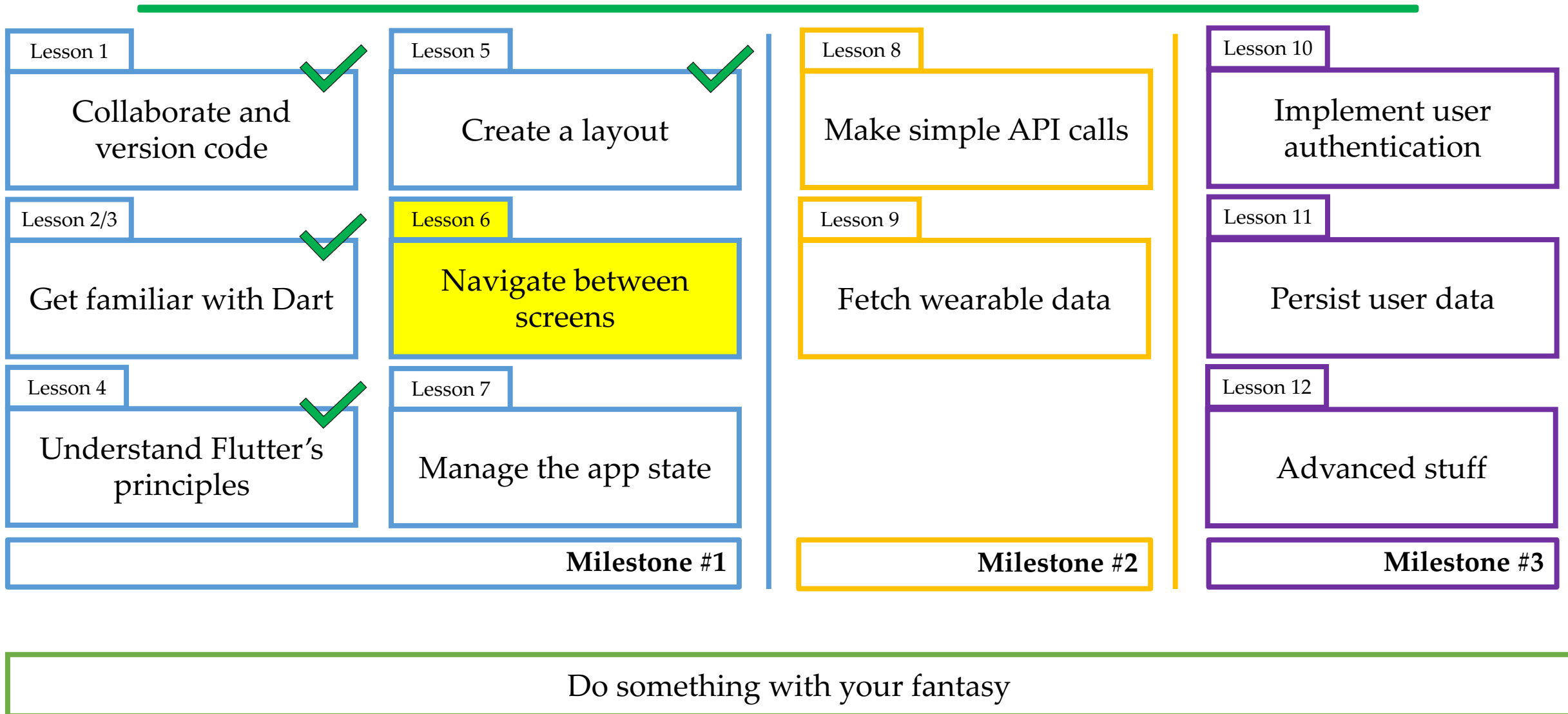


# Outline

---

- **Recap**
- Navigator
- Navigate to a new screen and back
- Named routes
- Passing argument to a named routes
- Returning an argument from a named route
  
- Exercise
- Homework
- Resources

# Recap



# Outline

---

- Recap
- **Navigator**
- Navigate to a new screen and back
- Named routes
- Passing argument to a named routes
- Returning an argument from a named route
- Exercise
- Homework
- Resources

# Navigator

---

- In general, apps are made of multiple screens (called **routes**)
- How to navigate through routes?
- How to pass things to routes and get values back from them?
- `Navigator` is a special class that allows to manage all of this

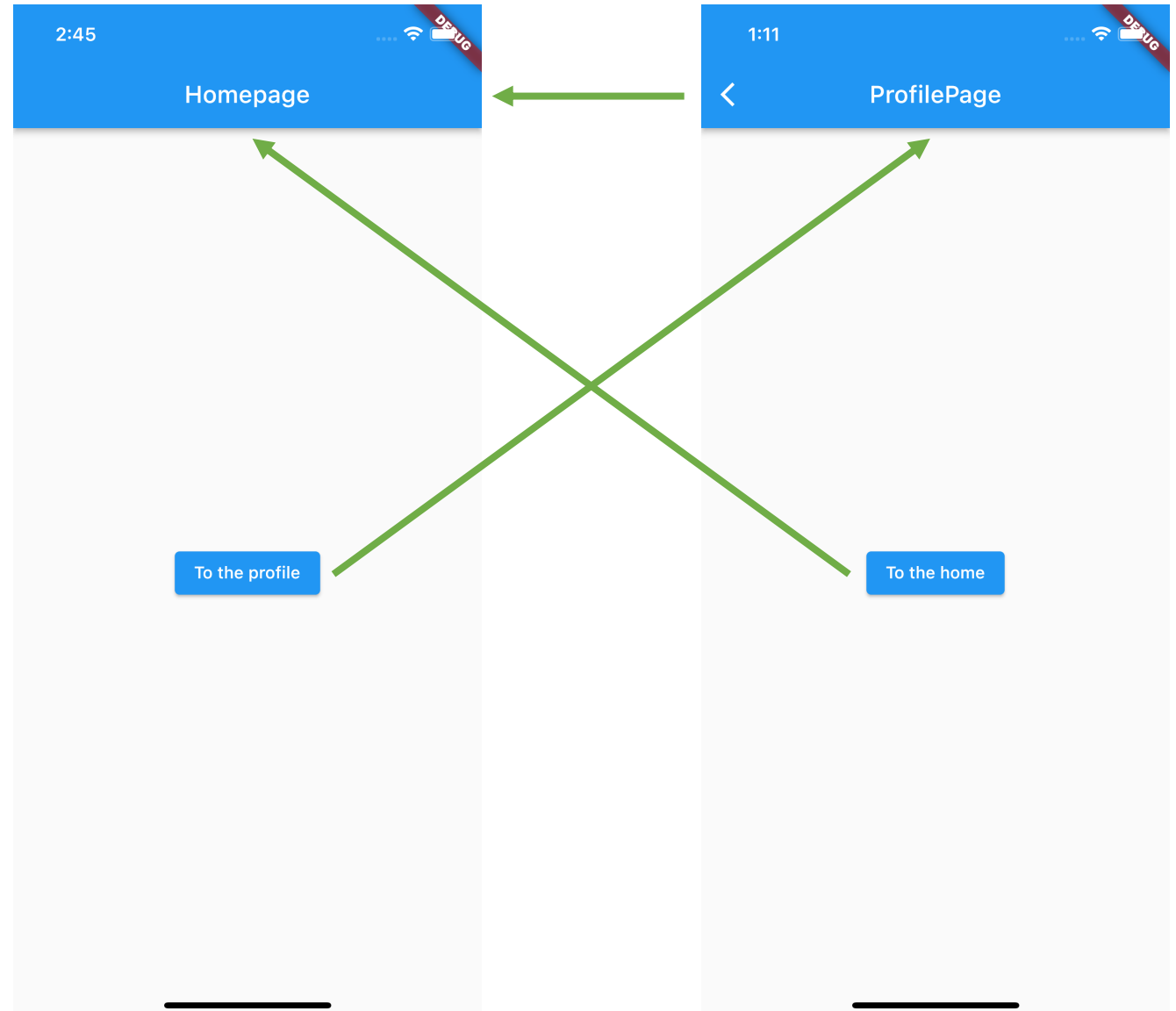
# Outline

---

- Recap
- Navigator
- **Navigate to a new screen and back**
- Named routes
- Passing argument to a named routes
- Returning an argument from a named route
  
- Exercise
- Homework
- Resources

# Navigator basics

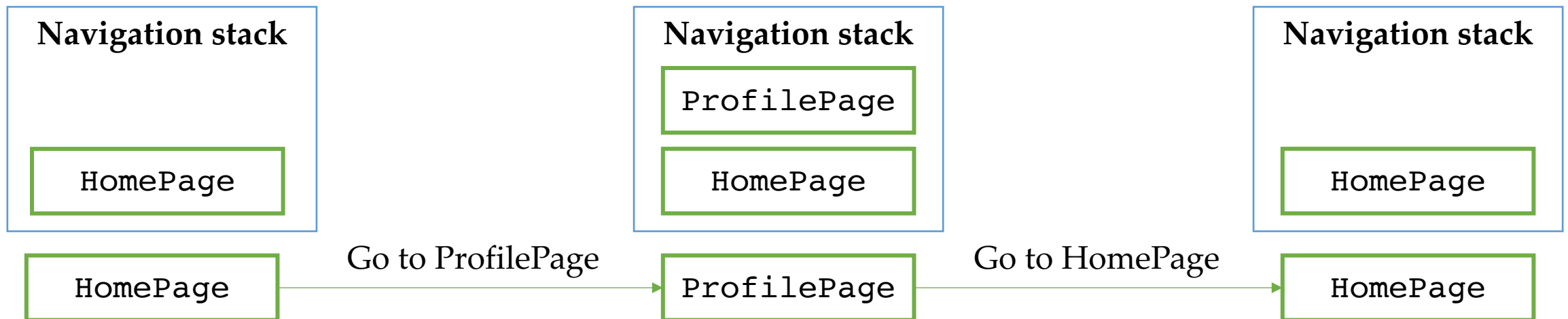
- First let's see how to move between two routes
- We will start from creating a simple two-routes app where the first route will act as homepage and the second will represent the route that will ideally contain the info on the user profile.
- When the user taps the button on the homepage it will be directed to the profile page and viceversa



# Navigator rationale

---

- Navigator is in charge of managing the navigation through the app
- To do so, Navigator uses a **stack-like structure**. The user sees the “top” of the stack
- When you go to a new route, you are “pushing” it into the stack
- When you go back, you are “popping” the route out of the Navigator

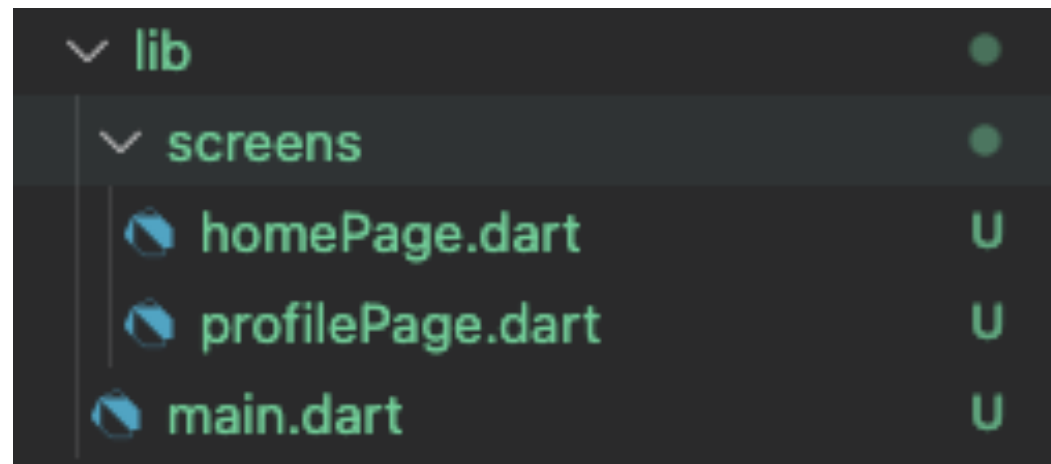




# Navigator basics - Preparation

---

- Create a new project called 'there\_and\_back\_again'
- Create the `lib/screens/` folder
- Create two files in the `lib/screens/` folder just created and rename them as 'homePage.dart' and 'profilePage.dart'
- The project `lib` folder should look like this:



# Navigator basics – homePage.dart boilerplate

---

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

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) :
    super(key: key);

  static const routename = 'Homepage';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(HomePage.routename),
      ),
      ...
    );
  }
}
```

```
...
body: Center(
  child: ElevatedButton(
    child: Text('To the profile'),
    onPressed: () {
      //TODO: implement the
      navigation
    },
  ),
);
} //build
} //HomePage
```

# Navigator basics – profilePage.dart boilerplate

---

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

class ProfilePage extends StatelessWidget {
  const ProfilePage({Key? key}) :
    super(key: key);

  static const routename = 'ProfilePage';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(HomePage.routename),
      ),
      ...

      ...
      body: Center(
        child: ElevatedButton(
          child: Text('To the home'),
          onPressed: () {
            //TODO: implement the
            navigation
          },
        ),
      ),
    );
  } //build
} //ProfilePage
```

# Navigator basics – main.dart boilerplate

---

```
import 'package:flutter/material.dart';
import
'package:there_and_back_again/screens/homepage.dart';

void main() {
  runApp(const MyApp());
} //main

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomePage(),
    );
  } //build
} //MyApp
```

# Navigator basics – push and pop

---

- To go to the ProfilePage route, simply invoke `Navigator.push()`:

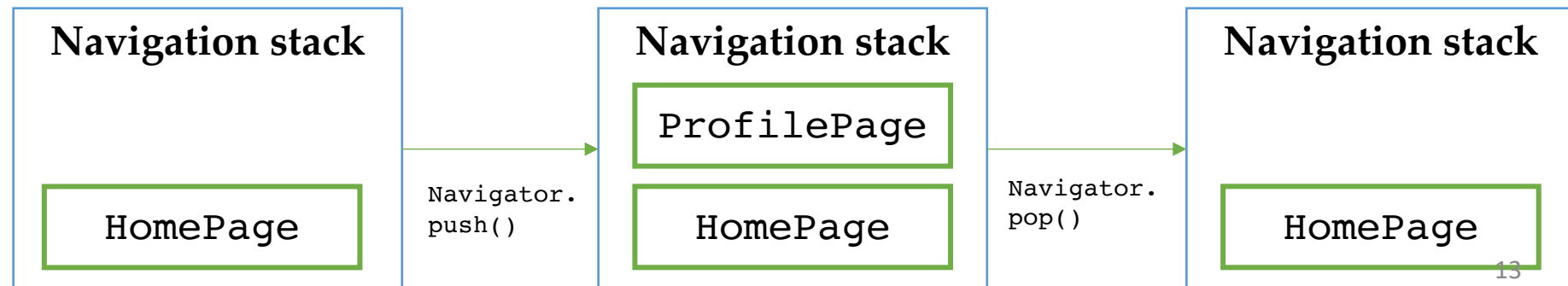
```
...  
onPressed: () {  
    Navigator.push(context, MaterialPageRoute(builder: (context) => ProfilePage()));  
},  
...
```

Current BuildContext

The new MaterialPageRoute to be pushed into the stack

- To pop the ProfilePage route, simply invoke `Navigator.pop()`:

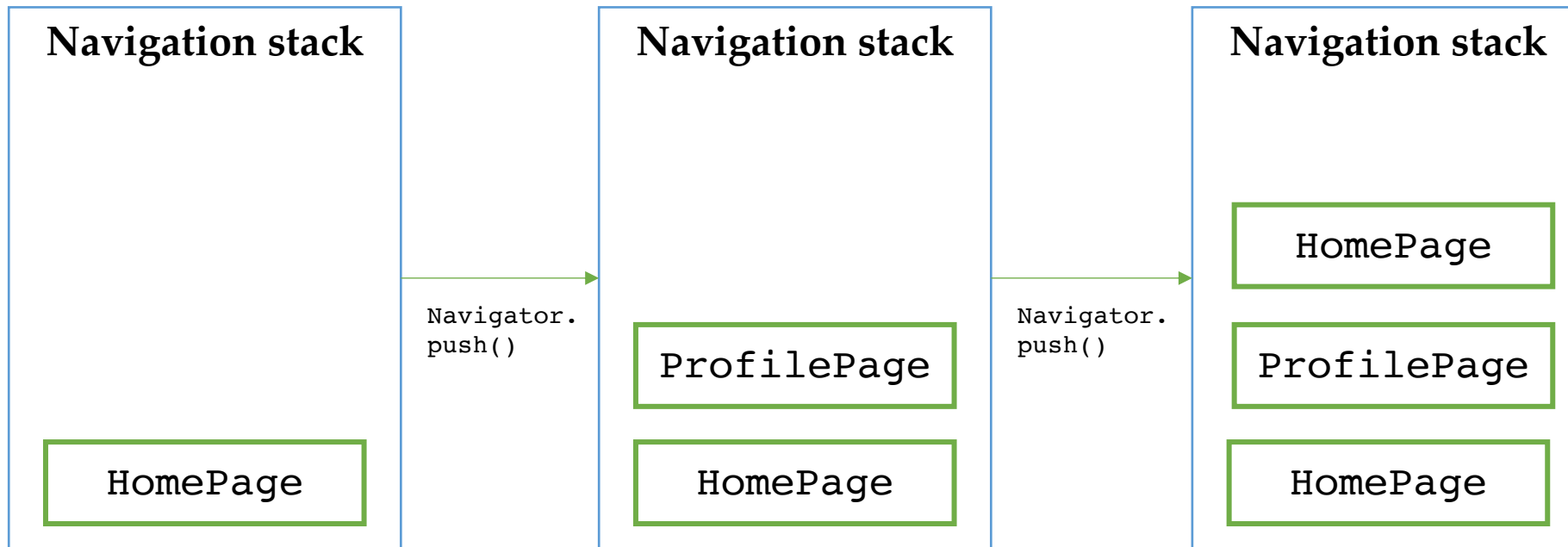
```
...  
onPressed: () {  
    Navigator.pop();  
},  
...
```



# Navigator basics – push and pop

---

- Note that you could have used `Navigator.push()` to go back to the `HomePage` but this would have been result:



- Very messy situation. The stack will grow indefinitely

# Outline

---

- Recap
- Navigator
- Navigate to a new screen and back
- **Named routes**
- Passing argument to a named routes
- Returning an argument from a named route
  
- Exercise
- Homework
- Resources

# Another approach: Named routes

---

- An alternative approach to `Navigator.push()` is `Navigator.pushNamed()`
- This solution consists of **associating names to each route** and use the names for navigation
- My personal opinion: this is a **cleaner approach** that leads to better, more readable code
- Let's see how to go for this approach



# Named navigation – Preparation

---

- If you want to implement this approach, you need to specify, from the beginning, the name of each route.
- This is done via the `initialRoute` and `routes` named parameters of `MaterialApp`:

```
MaterialApp(  
  initialRoute: '/',  
  routes: {  
    '/' : (context) => HomePage(),  
    '/profile/': (context) => ProfilePage(),  
  },  
);
```

This specifies the app entry point

This maps names to the corresponding routes within the app

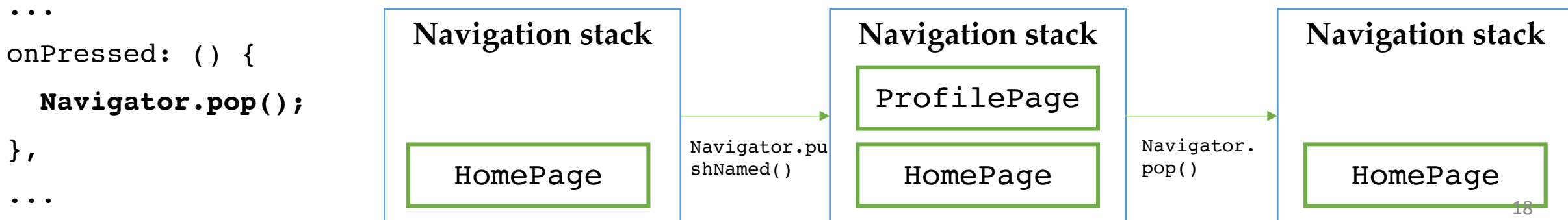
# Named navigation – pushNamed

- To go to the ProfilePage route, now you can invoke `Navigator.pushNamed()`:

```
...
onPressed: () {
  Navigator.pushNamed(context, '/profile/');
},
...
```

Current BuildContext      The name of the route to be pushed into the stack

- To pop the ProfilePage route, you can still use `Navigator.pop()`:



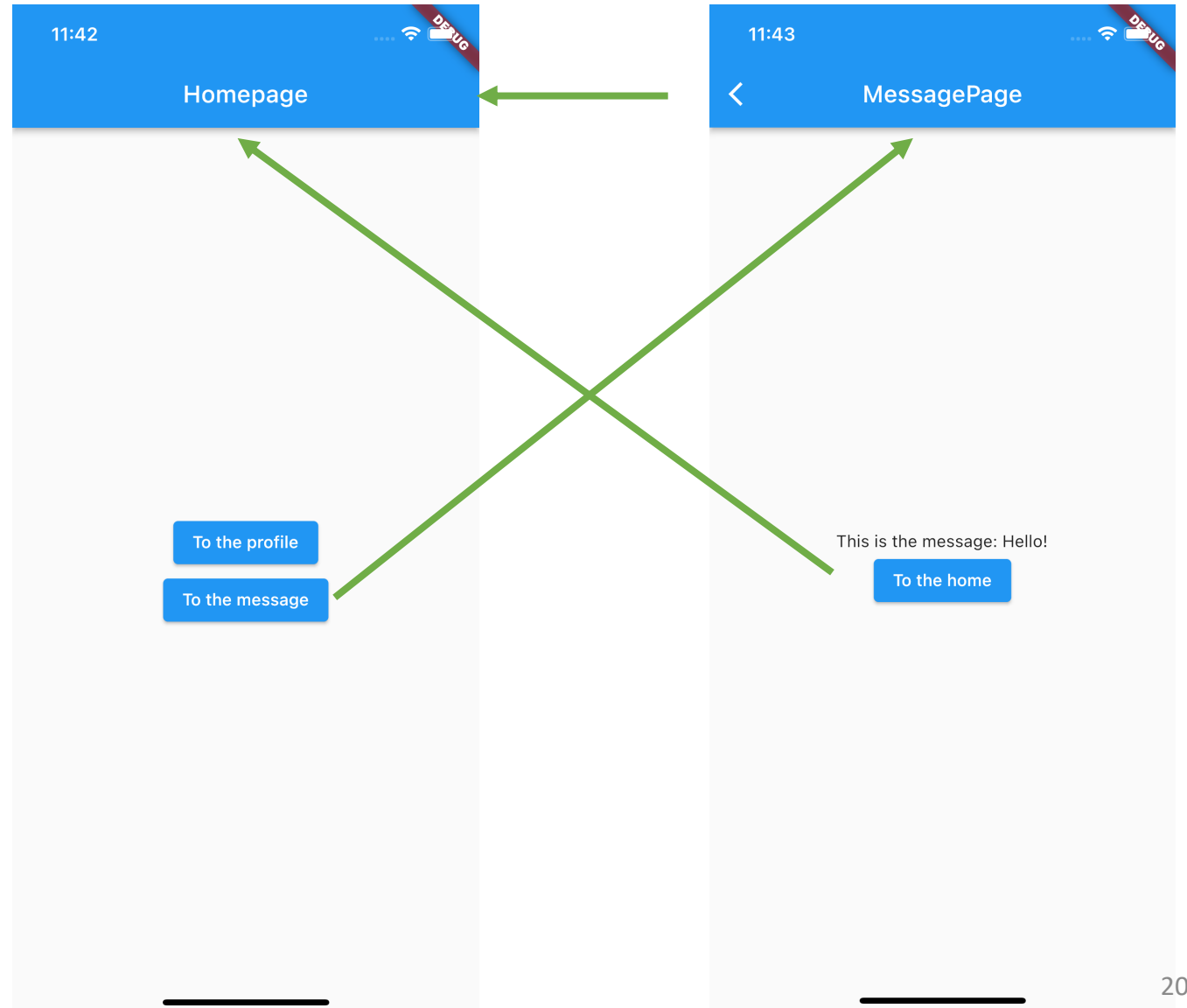
# Outline

---

- Recap
- Navigator
- Navigate to a new screen and back
- Named routes
- **Passing argument to a named routes**
- Returning an argument from a named route
- Exercise
- Homework
- Resources

# Navigator – Passing an argument

- It is (of course) possible to pass arguments to the new route that can be used for several purposes.
- To demonstrate how, let's expand the app with another route `MessagePage` that will get an argument from the `HomePage` and will show it in the center of the screen.



# Passing arguments – messagePage.dart boilerplate

---

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

class MessagePage extends StatelessWidget {
  const MessagePage({Key? key}) :
    super(key: key);

  static const routename = 'MessagePage';

  @override
  Widget build(BuildContext context) {
    //TODO: get the message from HomePage
    return Scaffold(
      appBar: AppBar(
        title: Text(MessagePage.routename),
      ),
      ...
    );
  }
}
```

```
...
body: Center(
  child: Column(
    mainAxisAlignment:
      MainAxisAlignment.center,
    children: [
      Text(''), //TODO: put the message
      inside the Text here
      ElevatedButton(
        child: Text('To the home'),
        onPressed: () {
          Navigator.pop(context);
        },
      ),
    ],
  ),
);

} //build
} // MessagePage
```

# Passing arguments – Add the new route and UI

---

- New route? Let's add it to the list:

```
MaterialApp(  
  initialRoute: '/',  
  routes: {  
    '/' : (context) => HomePage(),  
    '/profile/': (context) => ProfilePage(),  
     '/message/': (context) => MessagePage(),  
  },  
);
```

- To do: add a button in the HomePage to navigate to MessagePage

# Passing arguments

---

- To pass an argument to the `MessagePage` route, now you can invoke `Navigator.pushNamed()` as:

```
...  
onPressed: () {  
    Navigator.pushNamed(context, '/message/', arguments: 'Hello!');  
},  
...
```

Current BuildContext

The name of the route to be pushed into the stack

The arguments to be passed

- Note that you can pass ANYTHING as argument, not just a `String`.

# Retrieving arguments

---

- To retrieve the argument from the `MessagePage` route side you can use a `ModalRoute` as:

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

To figure out what this is, you can imagine that as a utility that stands between the previous route (here `HomePage`) and the current one (here `MessagePage`). For details see:

<https://api.flutter.dev/flutter/widgets/ModalRoute-class.html>

Arguments is an `Object`? But you know this is a `String`, so parse it explicitly!

We put the `!` here to force the non-null type.

- Then we display the retrieved argument by simply:

```
...  
Text('This is the message: $message'),  
...
```



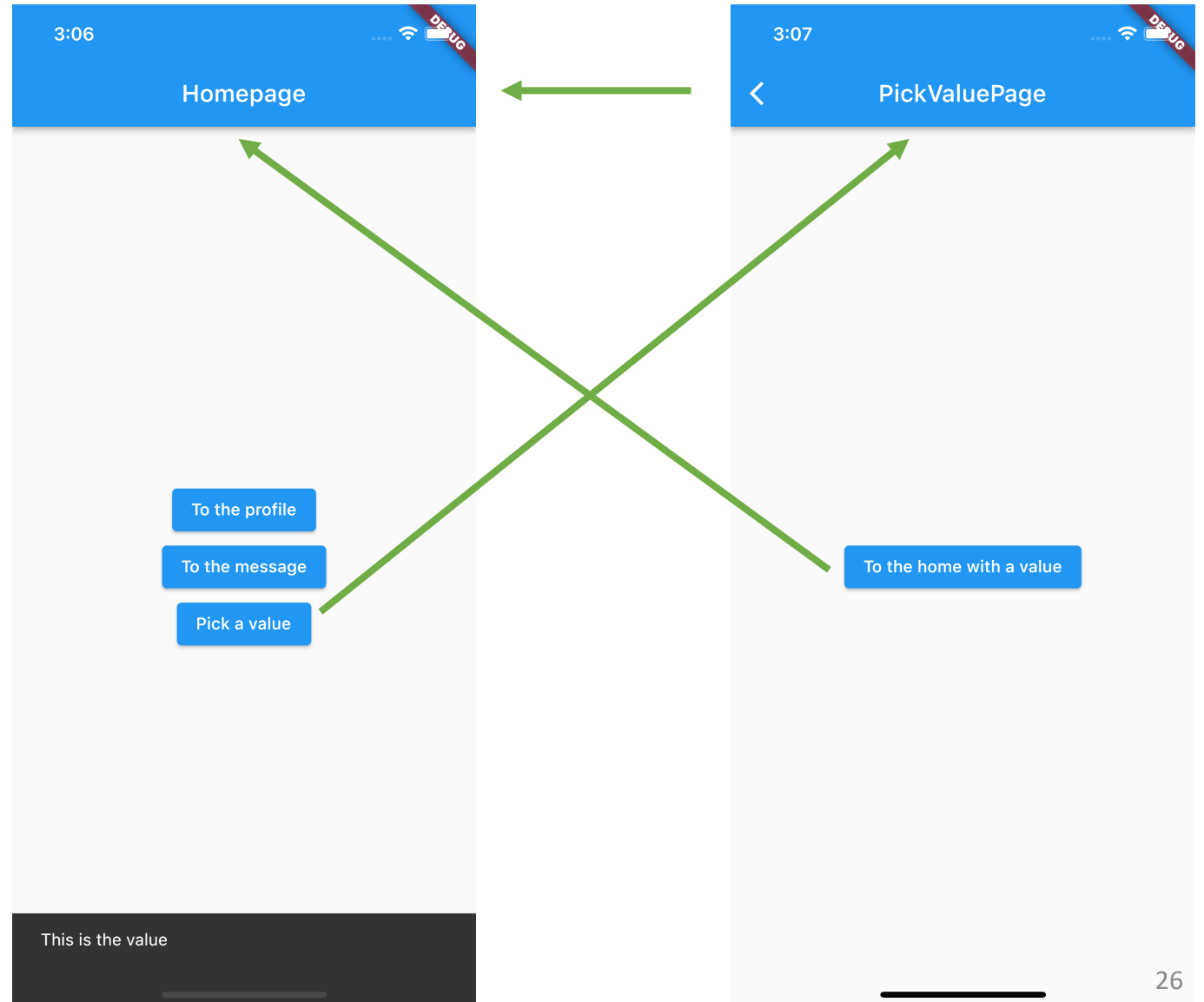
# Outline

---

- Recap
- Navigator
- Navigate to a new screen and back
- Named routes
- Passing argument to a named routes
- **Returning an argument from a named route**
- Exercise
- Homework
- Resources

# Navigator – Returning data

- It is (of course) also possible to return data from a route.
- To demonstrate how, let's expand the app with another route `PickValuePage` that will provide a value to the `HomePage` which will be in charge of showing it via a `ScaffoldMessenger`.



# Returning data – pickValuePage.dart boilerplate

---

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

class PickValuePage extends StatelessWidget {
  const PickValuePage({Key? key}) :
    super(key: key);

  static const routename = 'PickValuePage';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(PickValuePage.routename),
      ),
      ...
    );
  } //build
} //PickValuePage

...
body: Center(
  child: ElevatedButton(
    child: Text('To the home'),
    onPressed: () {
      //TODO: implement the
      navigation + return the data
    },
  ),
);
```

# Returning data – Add the new route

---

➤ New route? Let's add it to the list:

```
MaterialApp(  
  initialRoute: '/',  
  routes: {  
    '/' : (context) => HomePage(),  
    '/profile/': (context) => ProfilePage(),  
    '/message/': (context) => MessagePage(),  
     '/pickValue/': (context) => PickValuePage(),  
  },  
);
```

# Returning arguments

---

- To return an argument to the HomePage route, you can invoke `Navigator.pop()` as:

```
...  
onPressed: () {  
    Navigator.pop(context, 'This is the value');  
},  
...
```



The value that will return to the HomePage once PickValuePage is popped out from the stack

- Note that you can return ANYTHING, not just a `String`.

# Returning arguments

---

- To get the result, the HomePage must be patient and *await* for it::

Await means async stuff. The onPressed function become asynchronous as well so...

```
...
onPressed: () async {
  final result = await Navigator.pushNamed(context, '/pickValue/');
  ScaffoldMessenger.of(context)
    ..removeCurrentSnackBar()
    ..showSnackBar(SnackBar(content: Text('$result')));
},
...
```

# Outline

---

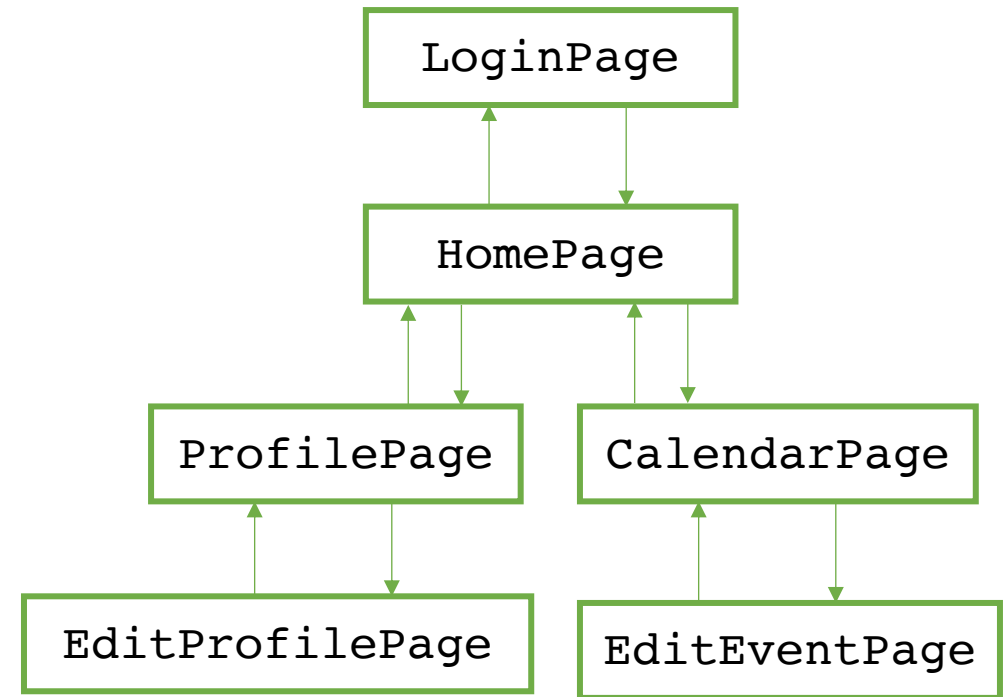
- Recap
- Navigator
- Navigate to a new screen and back
- Named routes
- Passing argument to a named routes
- Returning an argument from a named route
  
- **Exercise**
- Homework
- Resources

# Exercise

---

## ➤ Exercise 06.01 (easy)

- Create a new project 'reproduce\_structure'
- Reproduce the app navigation structure on the right using the **named routing approach**.



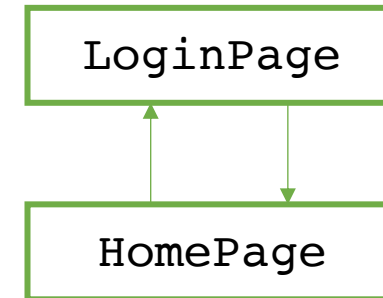


# Exercise

---

## ➤ Exercise 06.02 (medium)

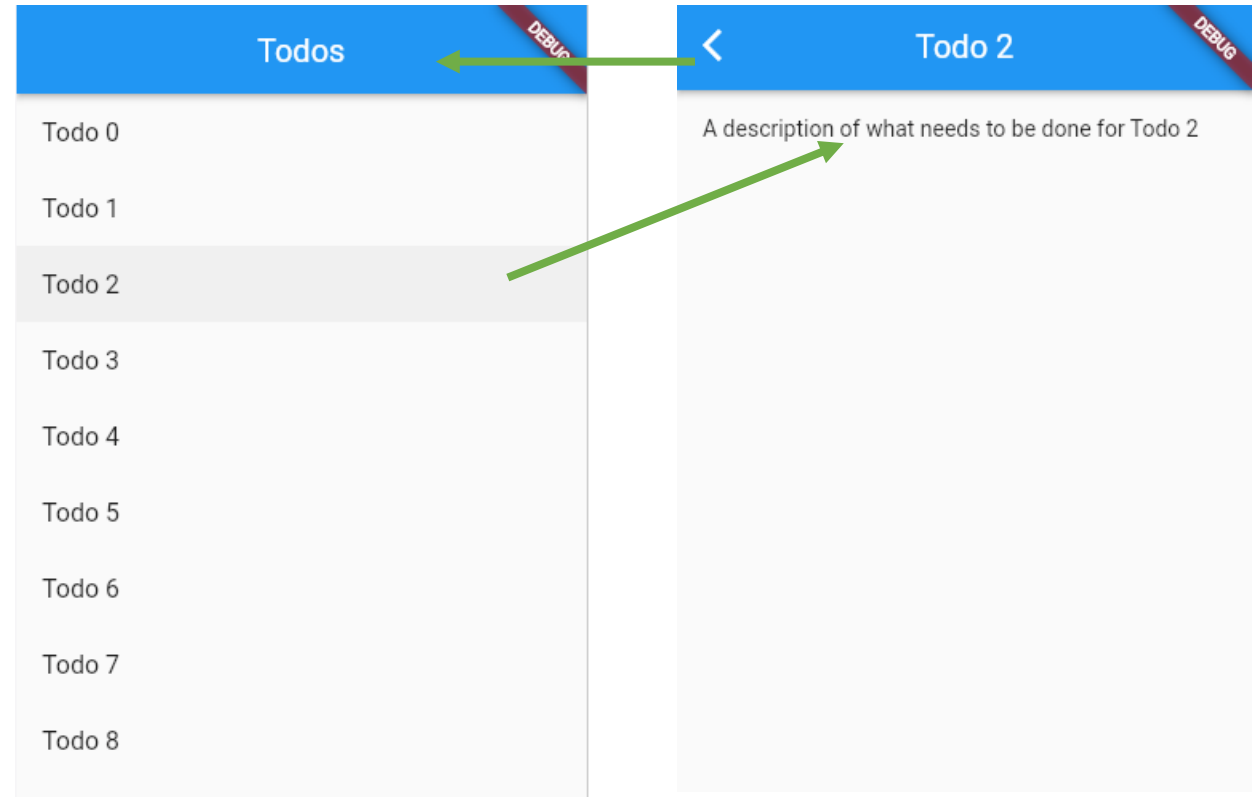
- Create a new project 'login\_flow'
- Reproduce the app navigation structure on the right using the named routing approach.
- The login page consists of a form with two textboxes (one for the username and the other for the password) and a button. Hint: you can use the widgets
- When the user types "bug@expert.com" in the username textbox and "5TrNgP5Wd" in the password textbox, and taps the button, the user is redirected to the Homepage. If the credentials are wrong, a `ScaffoldMessenger` is showed for 2 seconds saying "Wrong credentials".
- The `HomePage` must show the provided username.



# Exercise

## ➤ Exercise 06.03 (medium)

- Follow the cookbook <https://docs.flutter.dev/cookbook/navigation/passing-data> by the Flutter team to learn how to pass data to a route directly to its constructor.
- (solution available from the Flutter team in the cookbook)



# Outline

---

- Recap
- Navigator
- Navigate to a new screen and back
- Named routes
- Passing argument to a named routes
- Returning an argument from a named route
  
- Exercise
- **Homework**
- Resources

# Homework

---

- Get familiar with Navigator

# Outline

---

- Recap
- Navigator
- Navigate to a new screen and back
- Named routes
- Passing argument to a named routes
- Returning an argument from a named route
  
- Exercise
- Homework
- **Resources**

# Resources

---

- Navigation Recipes
  - <https://docs.flutter.dev/cookbook/navigation>