IEEE FCI - Helwan
Student Branch

S25' Training Sessions

MOBILE DEVELOPMENT

SESSION 8

#create_share_innovate

Table of contenet

Recap

Scrolling Widgets

Navigation and Routing

Responsive Widgets

Recap

1.Basic Widgets

Scaffold, AppBar, Text , Buttons , Icons, TextField

2.Layout Widgets

Column, Row, Container, SizedBox, Center

3.Assets

Images, Fonts, Video, Audio

Scrolling Widgets

What is Scrolling?

Scrolling allows users to navigate through content that doesn't fit on the screen.

Why is it important?

- Enhances user experience
- Essential for lists, feeds, forms, and more

Scrolling Widgets

Flutter provides built-in scrollable widgets:

- SingleChildScrollView
- ListView
- GridView

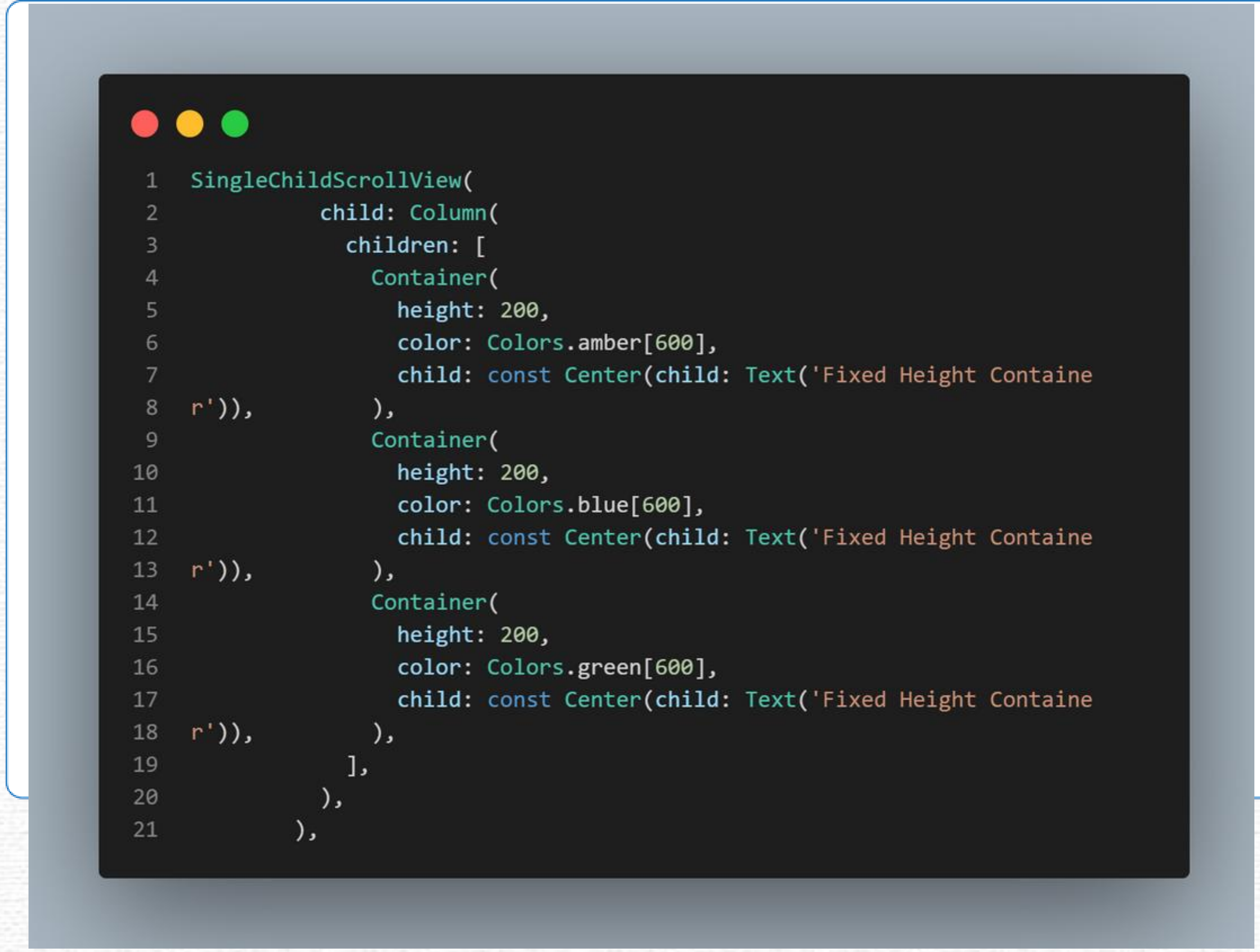
Scrolling Widgets

1- SingleChildScrollView

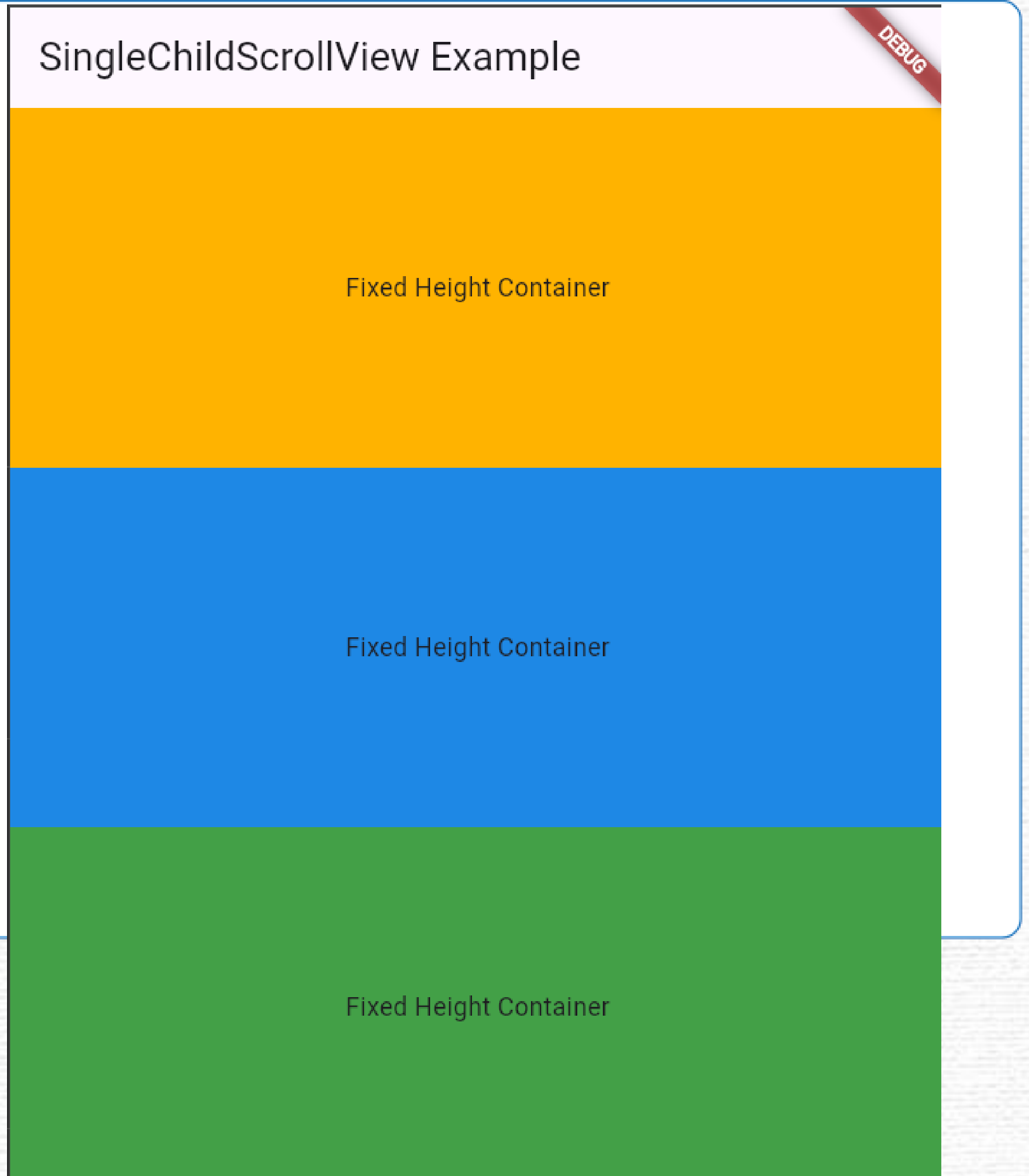
Use When:

- You have a small number of widgets
- You want vertical or horizontal scrolling

Scrolling Widgets



```
SingleChildScrollView(
     child: Column(
      children: [
       Container(
        height: 200,
        color: Colors.amber[600],
        child: const Center(child: Text('Fixed Height Containe
r')),  ),
       Container(
        height: 200,
        color: Colors.blue[600],
        child: const Center(child: Text('Fixed Height Containe
r')),  ),
       Container(
        height: 200,
        color: Colors.green[600],
        child: const Center(child: Text('Fixed Height Containe
r')),  ),
      ],
     ),
    ),
```



SingleChildScrollView Example

Fixed Height Container

Fixed Height Container

Fixed Height Container

Scrolling Widgets

2- ListView

Use When:

- You want to display a large or infinite number of items
- It handles scrolling efficiently

Types:

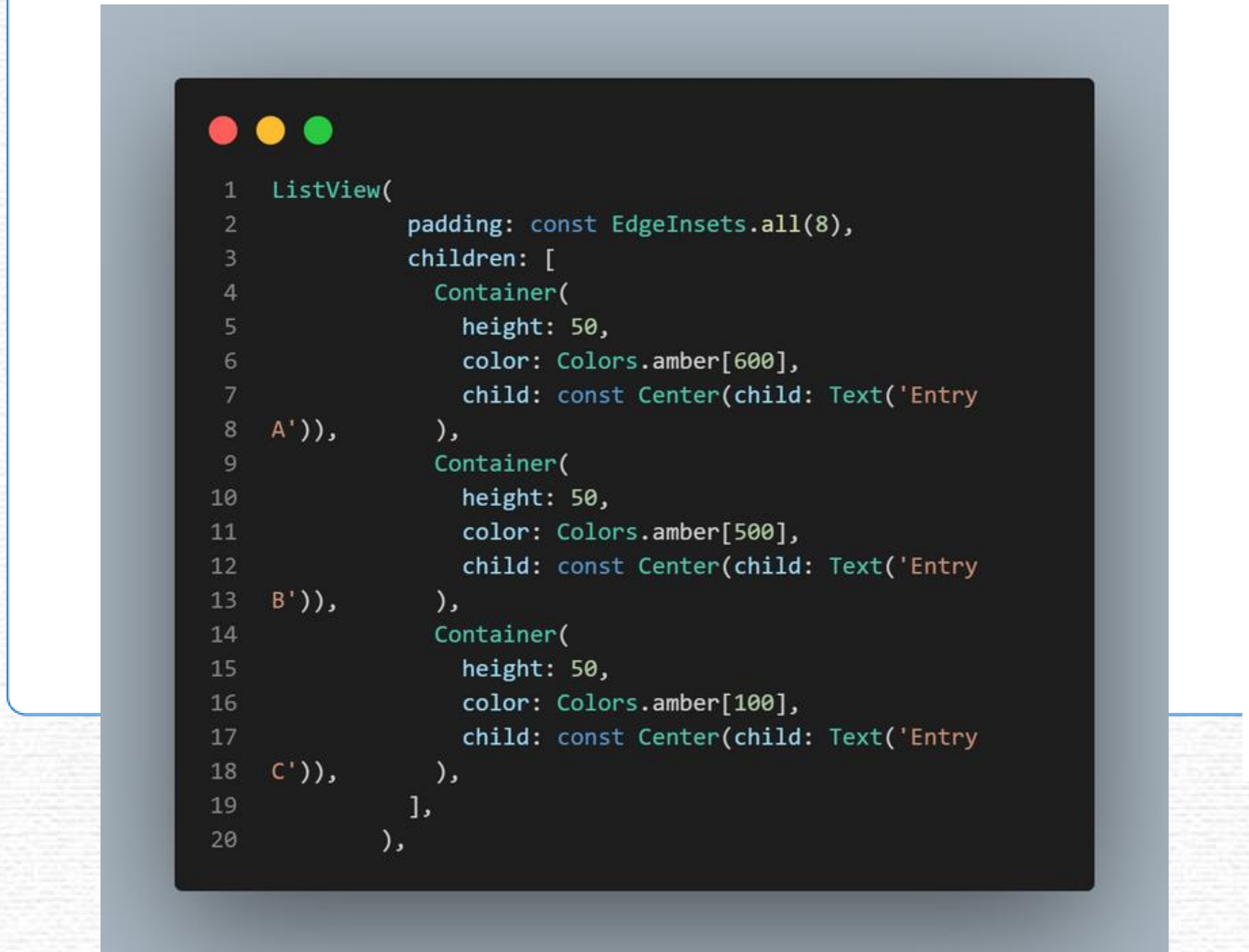
1.ListView()

2.ListView.builder()

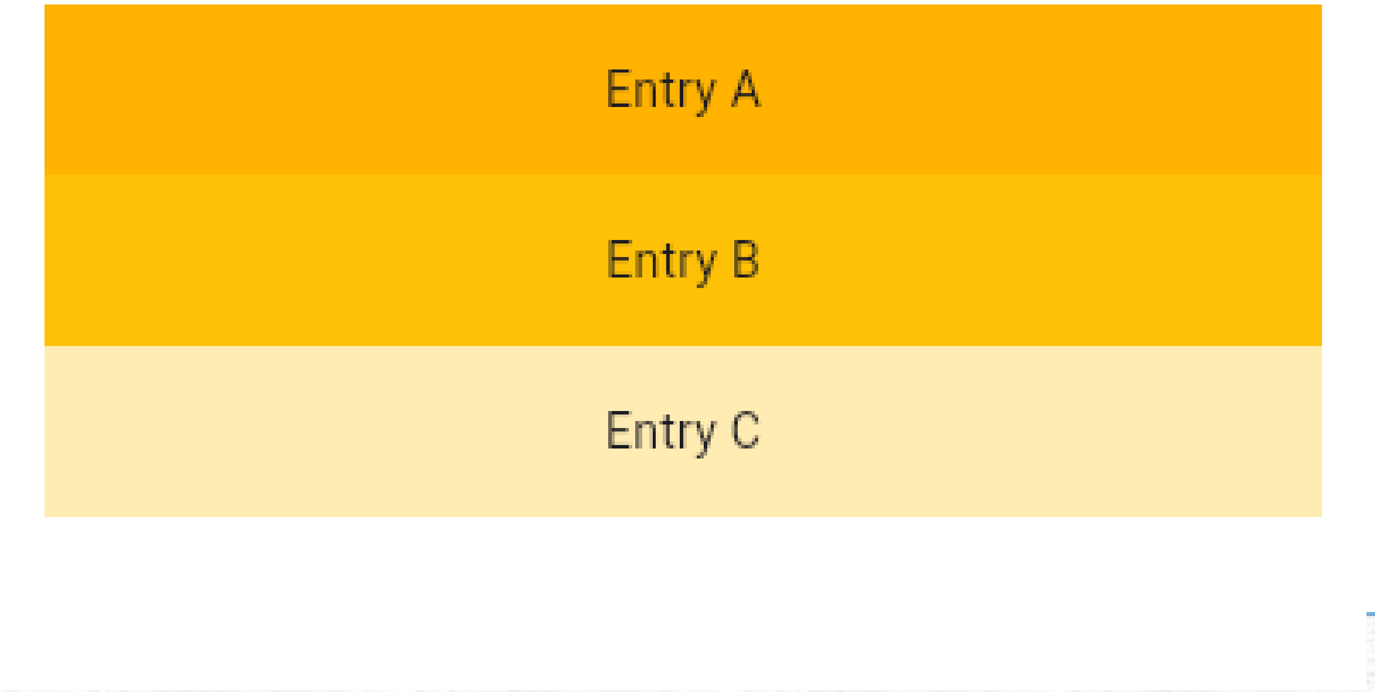
3.ListView.separated()

Scrolling Widgets

1-ListView()



```
ListView(
          padding: const EdgeInsets.all(8),
          children: [
            Container(
              height: 50,
              color: Colors.amber[600],
              child: const Center(child: Text('Entry
A')),       ),
            Container(
              height: 50,
              color: Colors.amber[500],
              child: const Center(child: Text('Entry
B')),       ),
            Container(
              height: 50,
              color: Colors.amber[100],
              child: const Center(child: Text('Entry
C')),       ),
          ],
        ),
```



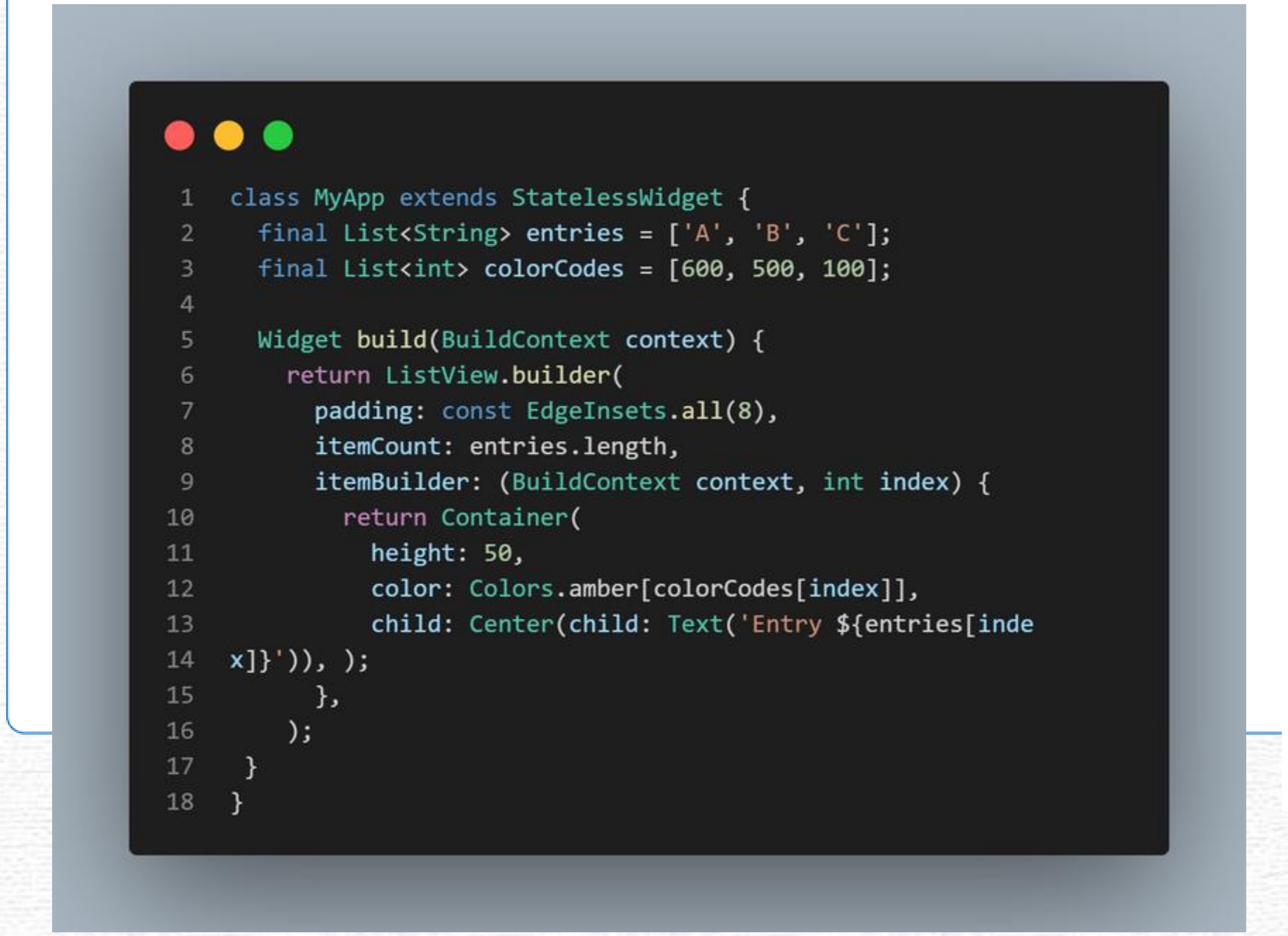
Entry A

Entry B

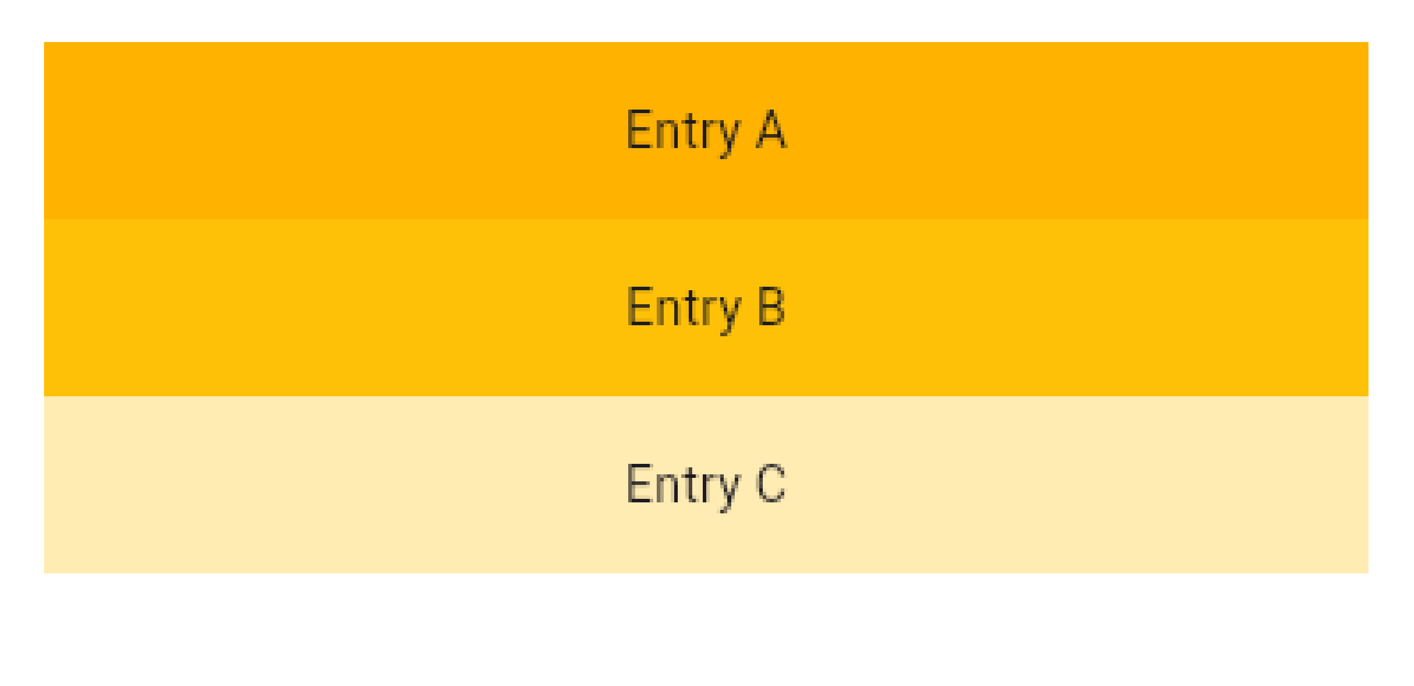
Entry C

Scrolling Widgets

2- ListView.builder()



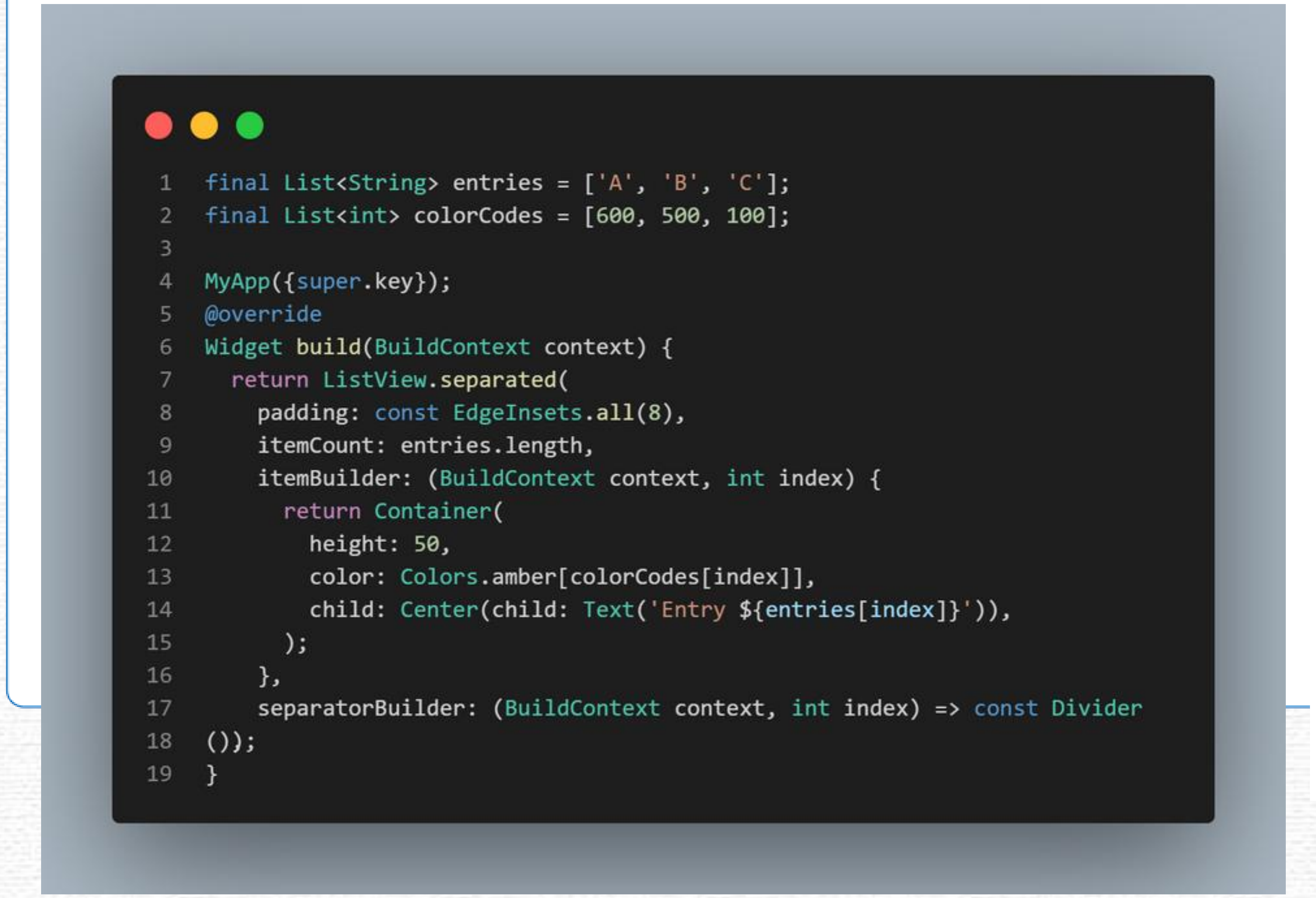
```
class MyApp extends StatelessWidget {
  final List<String> entries = ['A', 'B', 'C'];
  final List<int> colorCodes = [600, 500, 100];

  Widget build(BuildContext context) {
    return ListView.builder(
      padding: const EdgeInsets.all(8),
      itemCount: entries.length,
      itemBuilder: (BuildContext context, int index) {
        return Container(
          height: 50,
          color: Colors.amber[colorCodes[index]],
          child: Center(child: Text('Entry ${entries[inde
x]}')), );
      },
    );
  }
}
```



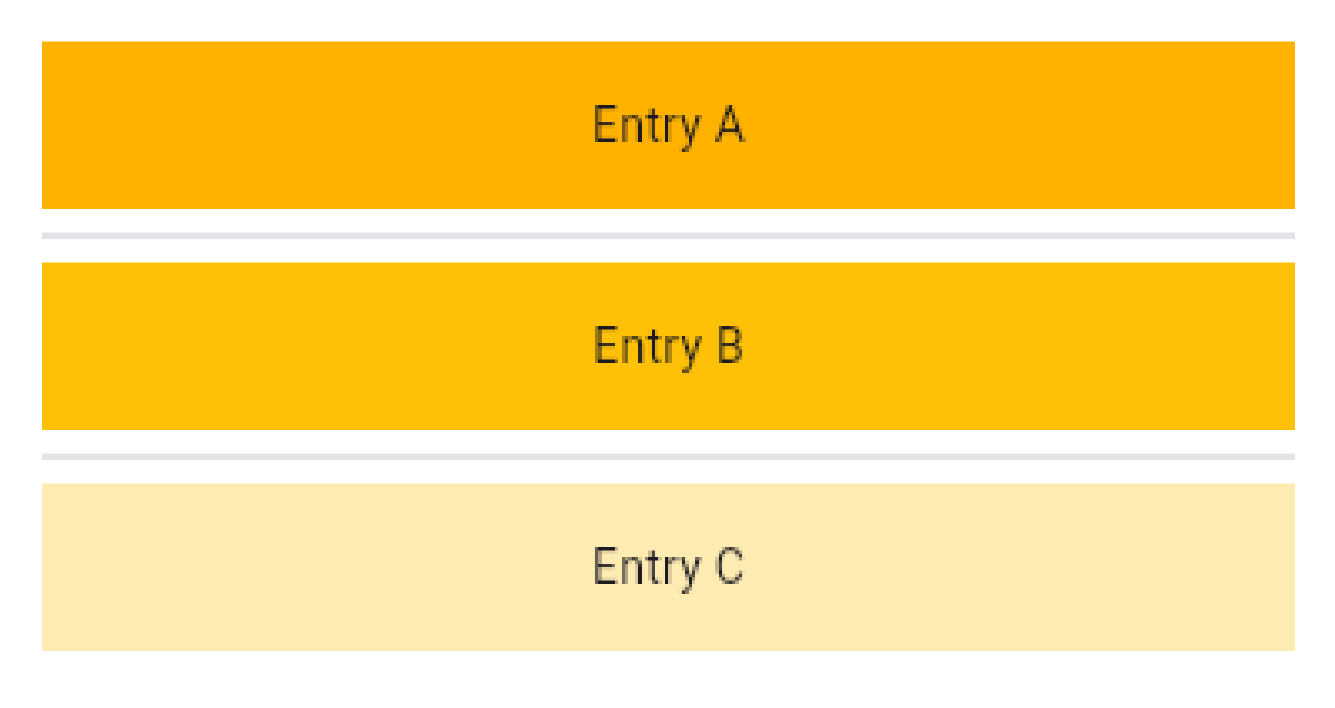
```
Entry A
Entry B
Entry C
```

Scrolling Widgets

3- ListView.separated()



```
final List<String> entries = ['A', 'B', 'C'];
final List<int> colorCodes = [600, 500, 100];

MyApp({super.key});
@override
Widget build(BuildContext context) {
  return ListView.separated(
    padding: const EdgeInsets.all(8),
    itemCount: entries.length,
    itemBuilder: (BuildContext context, int index) {
      return Container(
        height: 50,
        color: Colors.amber[colorCodes[index]],
        child: Center(child: Text('Entry ${entries[index]}')),
      );
    },
    separatorBuilder: (BuildContext context, int index) => const Divider
());
}
```



Entry A

Entry B

Entry C

Scrolling Widgets

3-GridView

Use When:

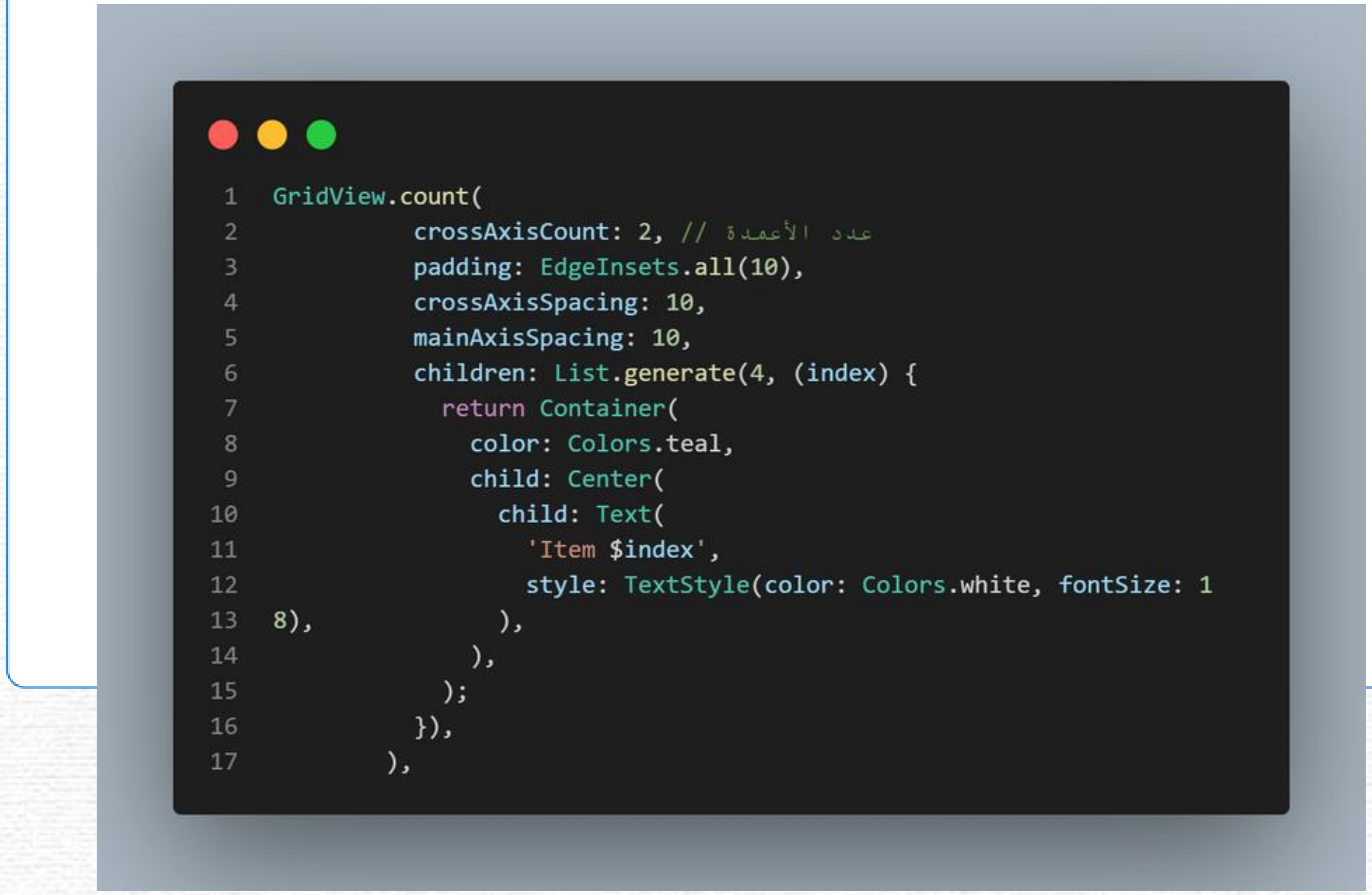
You want to display items in a grid format

Types:

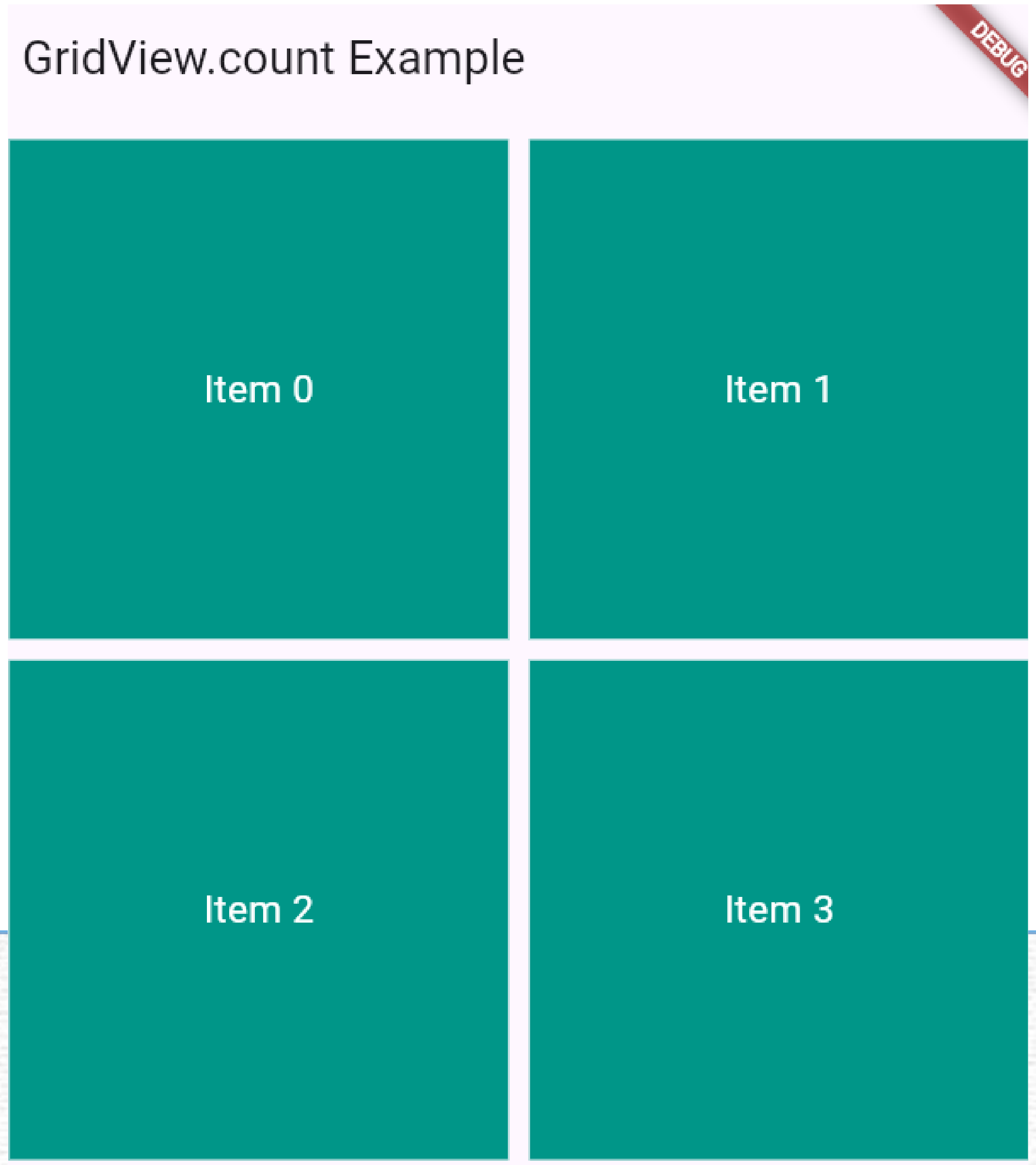
1. **GridView.count()**
2. **GridView.builder()**

Scrolling Widgets

1- GridView.count()



```
GridView.count(
          crossAxisCount: 2, // عدد الأعمدة
          padding: EdgeInsets.all(10),
          crossAxisSpacing: 10,
          mainAxisSpacing: 10,
          children: List.generate(4, (index) {
            return Container(
              color: Colors.teal,
              child: Center(
                child: Text(
                  'Item $index',
                  style: TextStyle(color: Colors.white, fontSize: 1
8),             ),
              ),
            );
          }),
        ),
```



GridView.count Example

Item 0

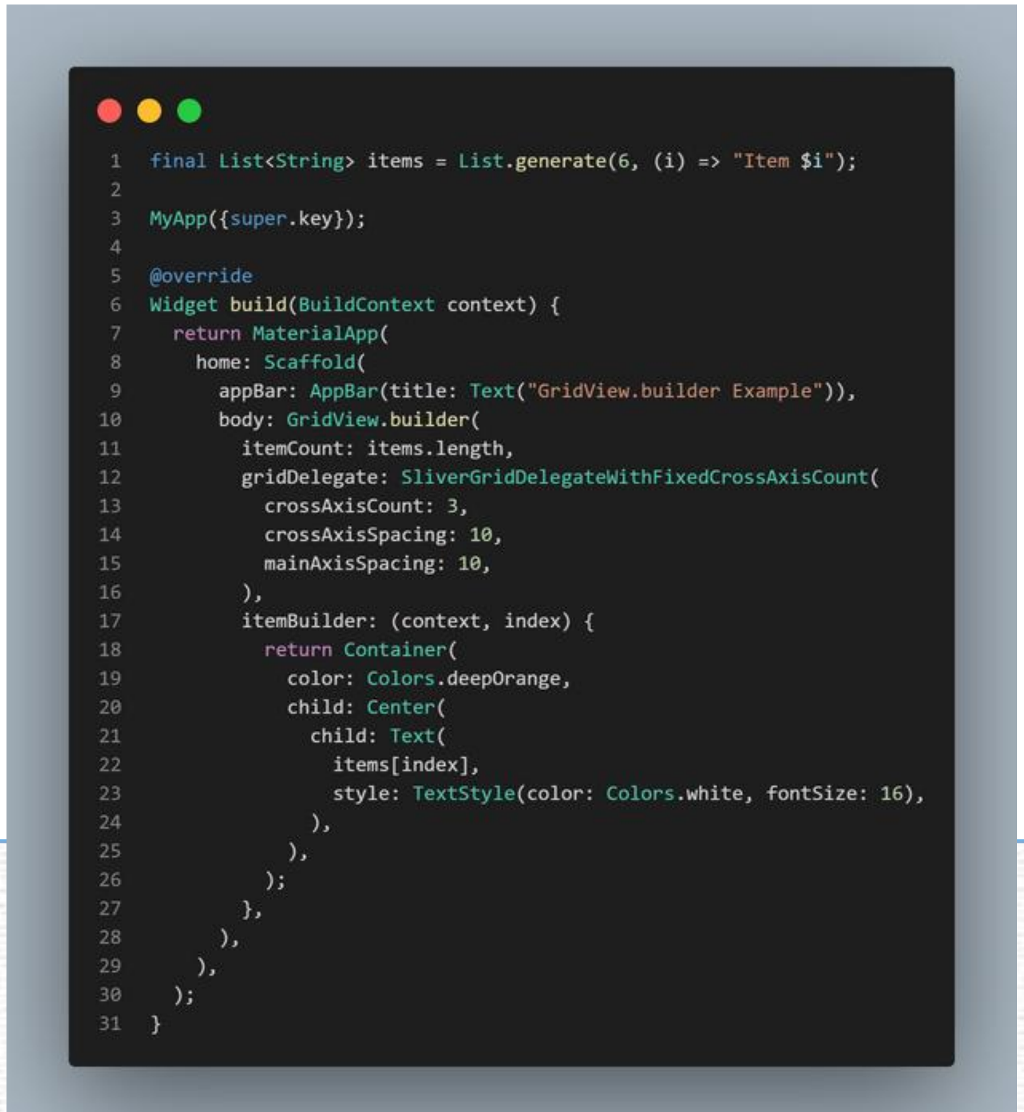
Item 1

Item 2

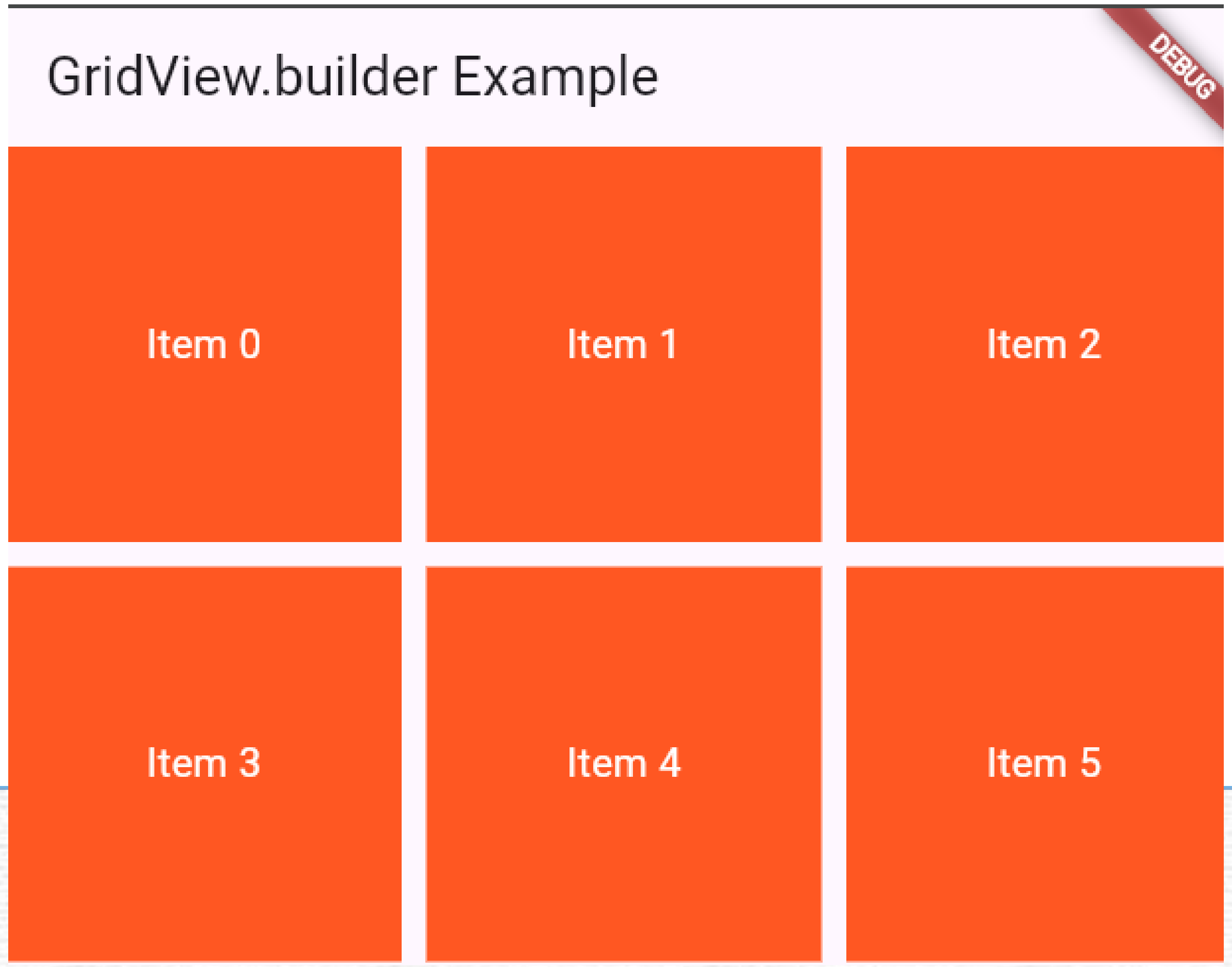
Item 3

Scrolling Widgets

2- GridView.builder()



```
final List<String> items = List.generate(6, (i) => "Item $i");

MyApp({super.key});

@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: Text("GridView.builder Example")),
      body: GridView.builder(
        itemCount: items.length,
        gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 3,
          crossAxisSpacing: 10,
          mainAxisSpacing: 10,
        ),
        itemBuilder: (context, index) {
          return Container(
            color: Colors.deepOrange,
            child: Center(
              child: Text(
                items[index],
                style: TextStyle(color: Colors.white, fontSize: 16),
              ),
            ),
          );
        },
      ),
    ),
  );
}
```



GridView.builder Example

Item 0 Item 1 Item 2

Item 3 Item 4 Item 5

Navigation & Routing

What is Navigation & Routing?

Navigation: Moving between different screens (pages).

Routing: Managing the route (path) to each screen.

⚠️ **In Flutter, screens are called Widgets, usually built as StatelessWidget or StatefulWidget.** ⚠️

Navigation & Routing

Why Use Navigation?

- Apps need multiple pages (Home, Settings, Profile).
- Enables a smooth user experience.
- Makes the app organized and modular.

Navigation & Routing

Basic Navigation:

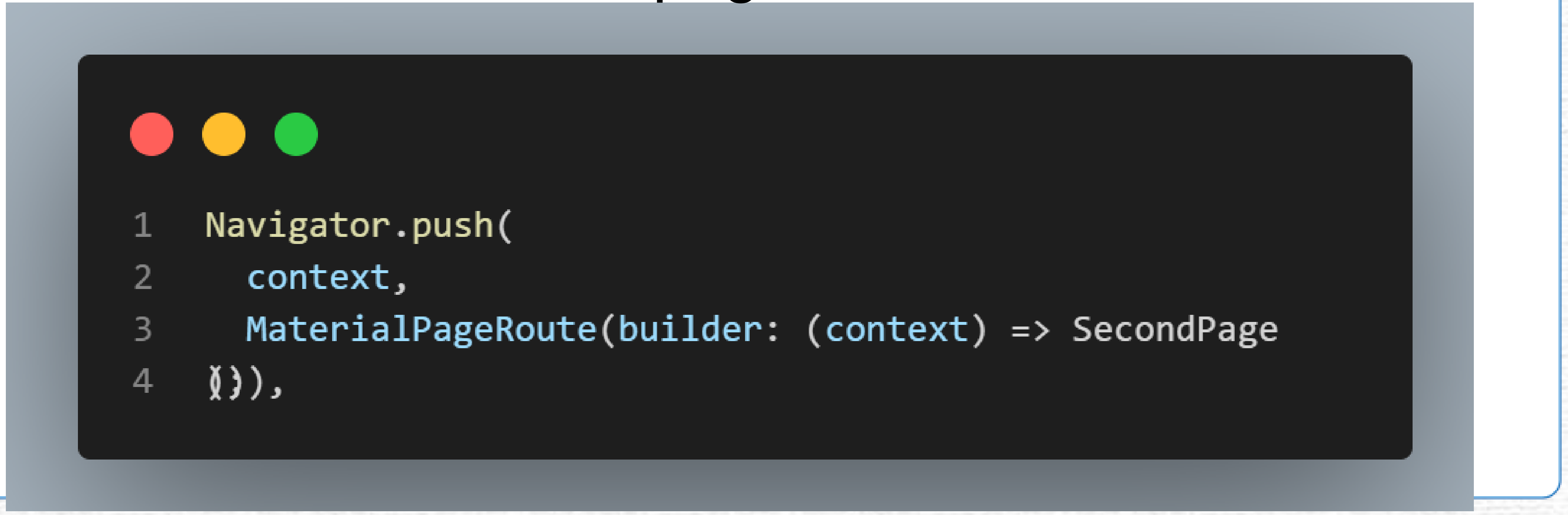
1- **Navigator.push():** Navigating to a new screen

2- **Navigator.pop():** Going back to the previous screen

Navigation & Routing

1.Navigating to a new screen(PUSH)

- Pushes SecondPage onto the navigation stack.
- MaterialPageRoute creates a transition between pages.



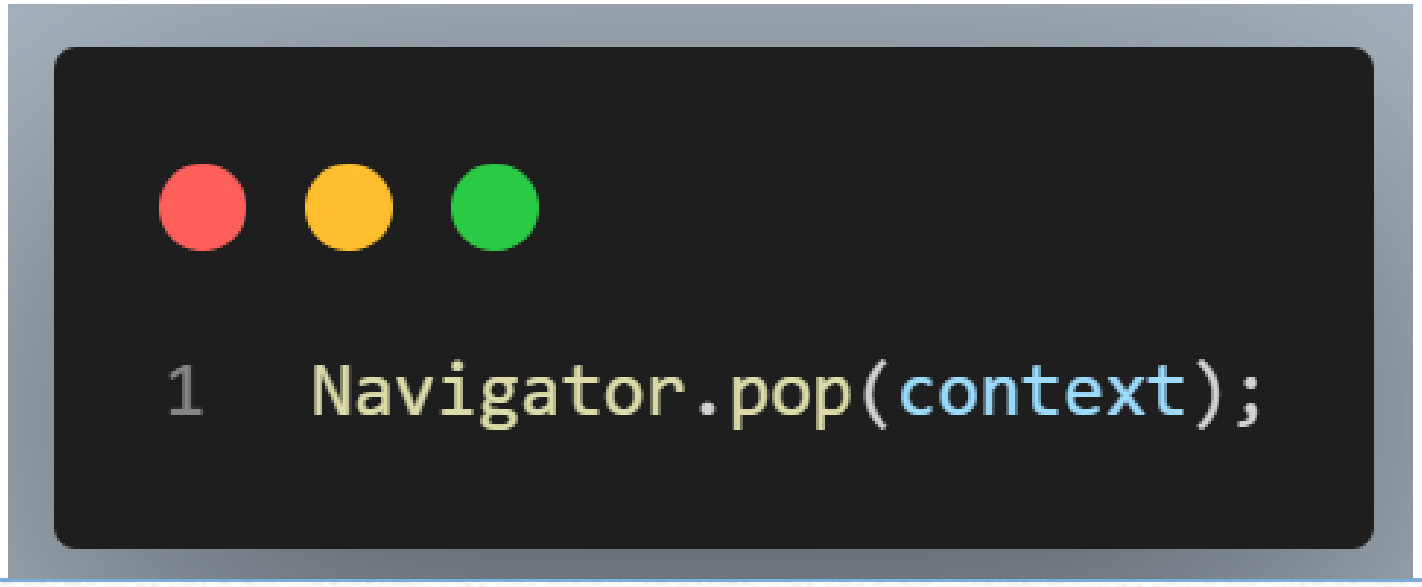
```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => SecondPage
()}),
```

S25' Training Sessions

Navigation & Routing

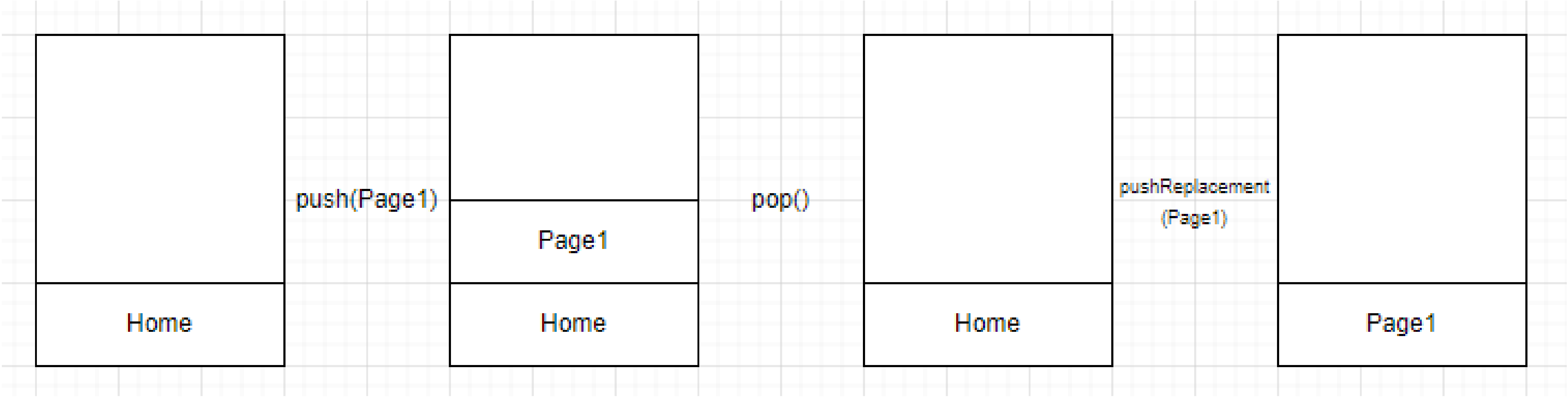
2.Going Back to the Previous Screen (POP):

- Removes the current page from the stack.
- Returns to the previous screen.



```dart
Navigator.pop(context);
```

Navigation & Routing



Home

push(Page1)

Page1

Home

pop()

Home

pushReplacement (Page1)

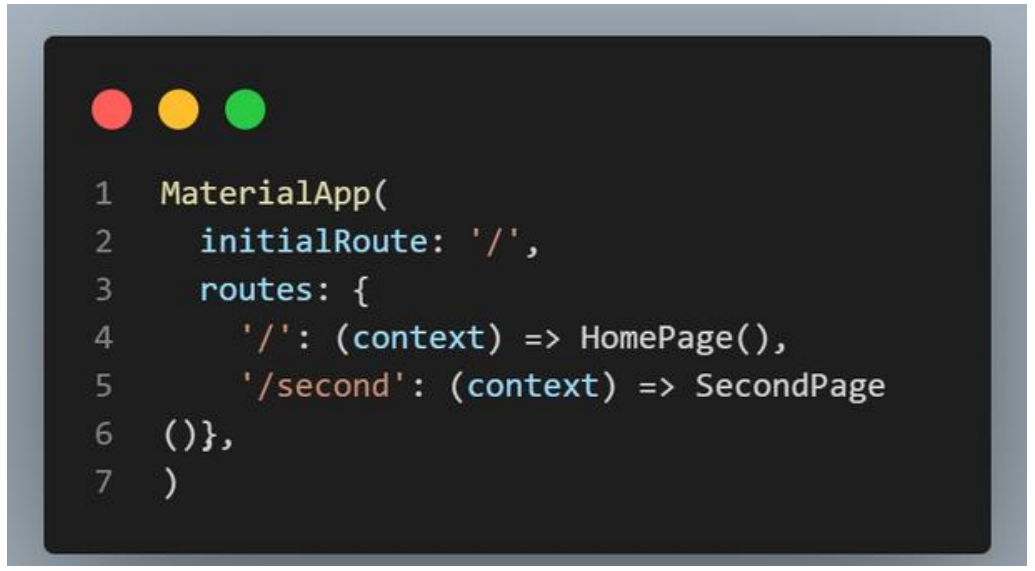
Page1

Navigation & Routing

Named Routes

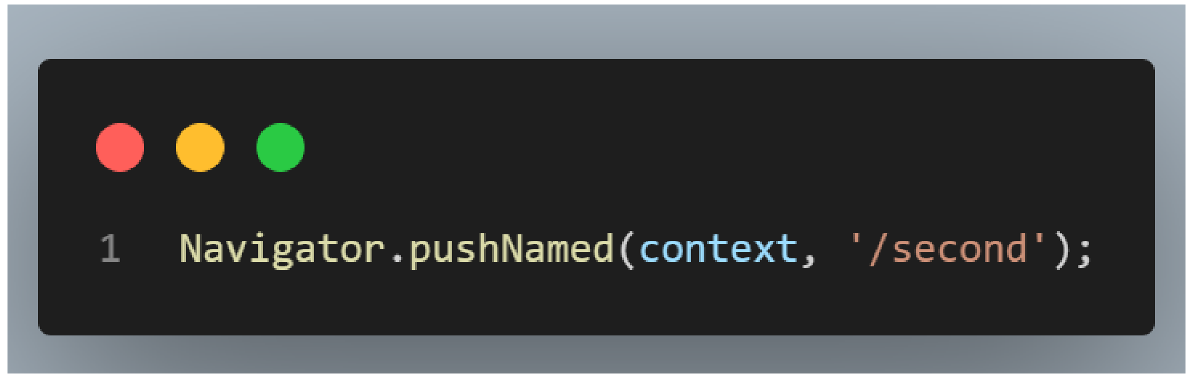
Better for managing multiple screens.

1-Define routes in MaterialApp:



```
MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => HomePage(),
    '/second': (context) => SecondPage
()},
)
```

2-Navigate using:

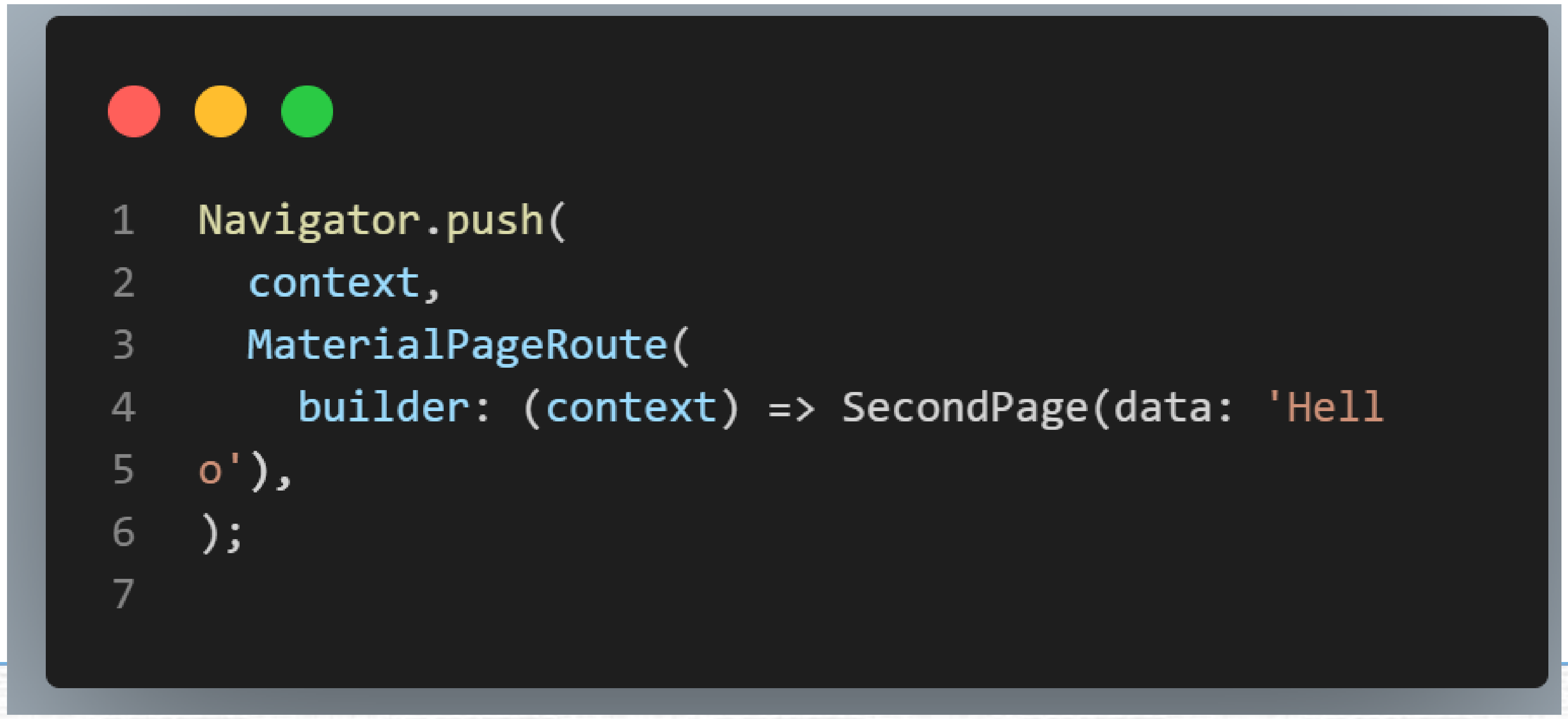


```
Navigator.pushNamed(context, '/second');
```

Navigation & Routing

Passing Data Between Screens

Passing Data with Navigator.push()

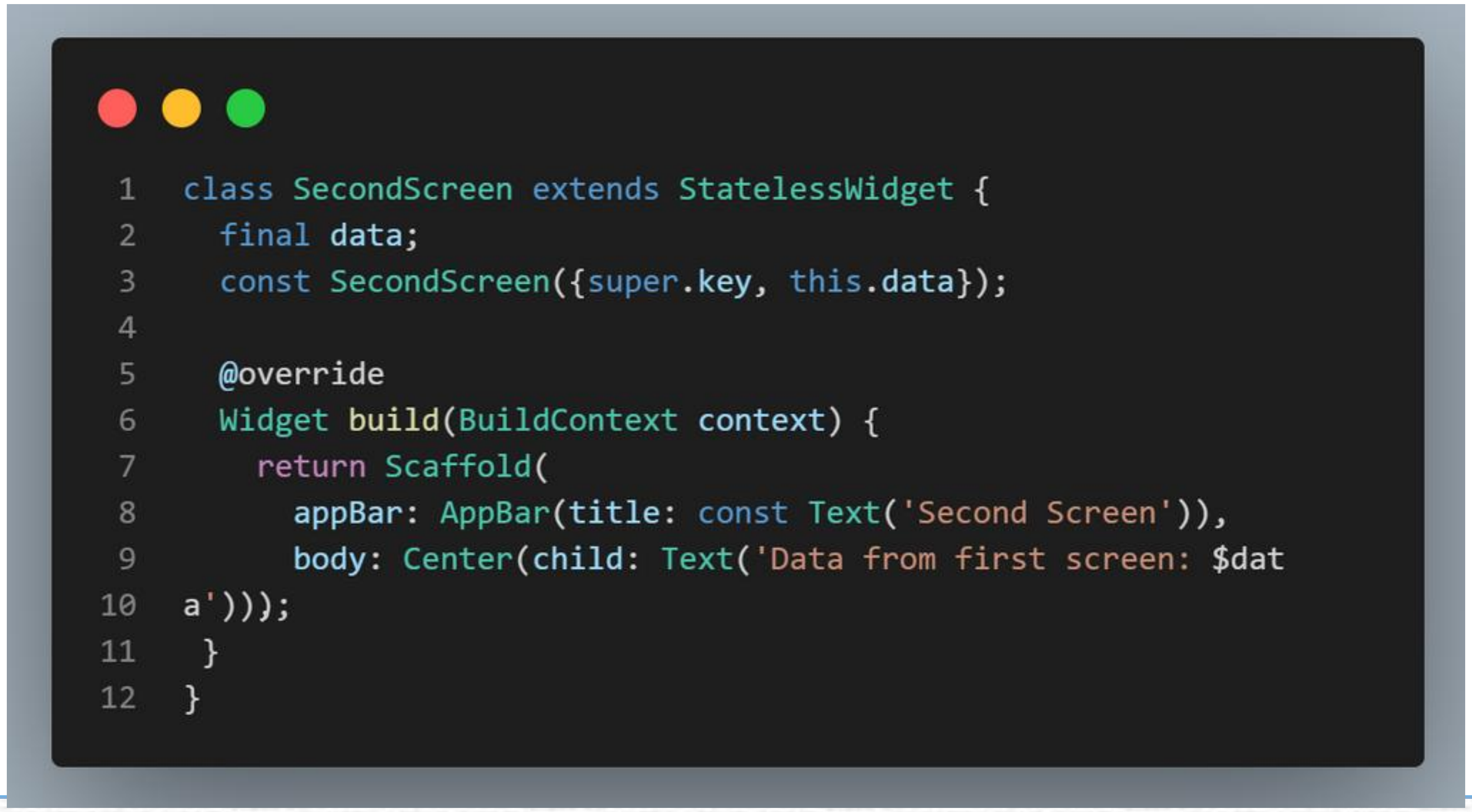


```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => SecondPage(data: 'Hell
o'),
);

```

S25' Training Sessions

Navigation & Routing

Receive the data in SecondScreen



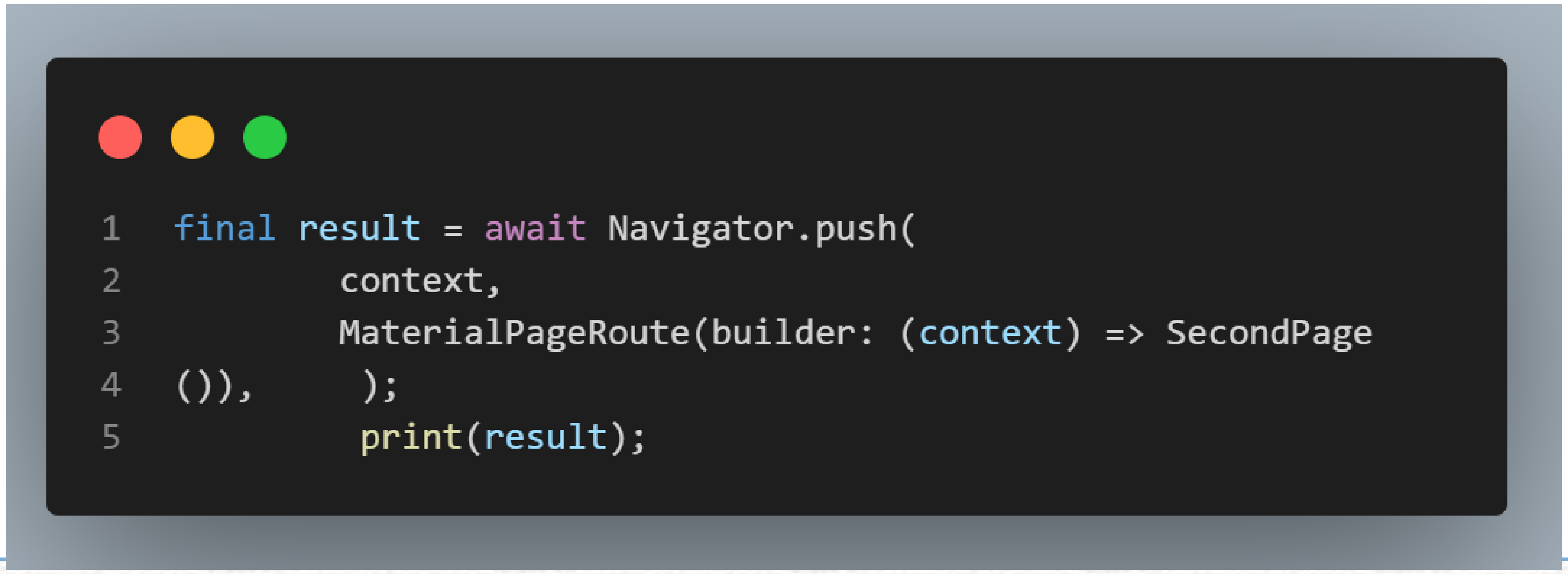
```dart
class SecondScreen extends StatelessWidget {
  final data;
  const SecondScreen({super.key, this.data});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(child: Text('Data from first screen: $dat
a')));
 }
}
```

Navigation & Routing

Pop Method

Returning Data with Navigator.pop()

Used to send data back to the previous screen.

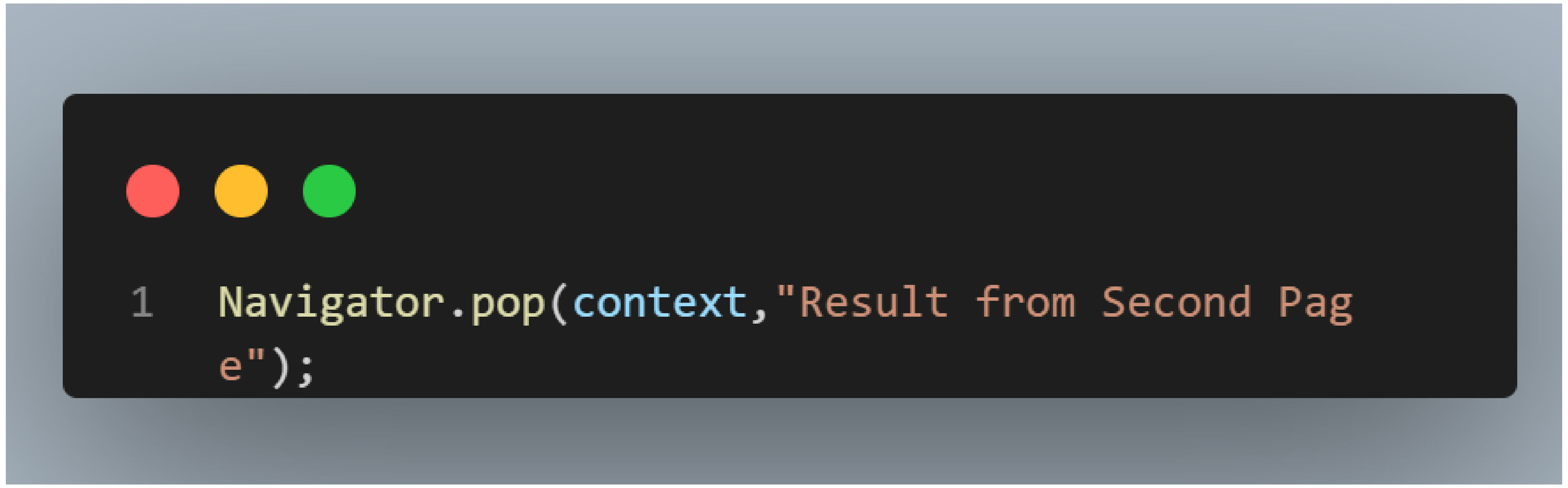


```
final result = await Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => SecondPage
()),     );
         print(result);
```

S25' Training Sessions

Navigation & Routing

Return data from SecondScreen:



```
Navigator.pop(context,"Result from Second Page");
```

Exercise

Create a basic Flutter app with two screens where:

1-Screen 1 (HomeScreen) has a button to navigate to Screen 2 (SecondScreen) and send a message using Navigator.push().

2- Screen 2 (SecondScreen) displays the received message and has a button to return a response using Navigator.pop().

3- The returned data is displayed on Screen 1 when the user comes back.

Responsive Widgets

What is Responsive UI?

- UI that adapts to different screen sizes & orientations.
- Essential for supporting phones, tablets, desktops, etc.

How to Achieve Responsive

1- Use MediaQuery – Get screen dimensions dynamically

2- Use Flexible Layout Widgets – Expanded, Flexible, etc.

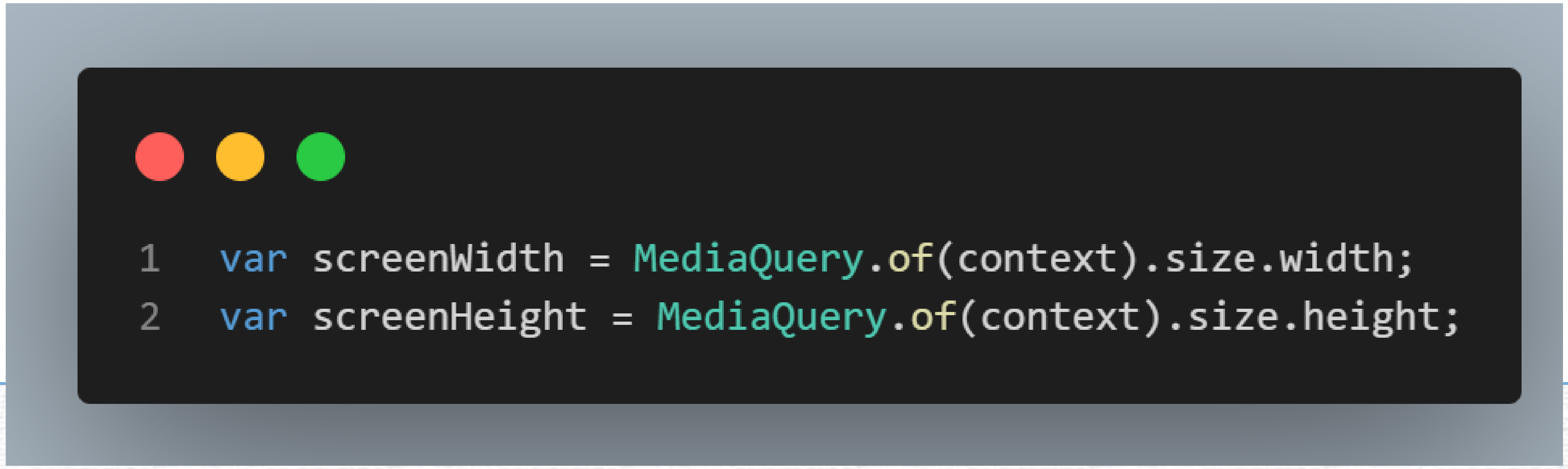
3- Use Aspect Ratio & Intrinsic Dimensions

Responsive Widgets

1- MediaQuery (Get Screen Dimensions Dynamically)

What is MediaQuery?

A way to get device size, orientation, padding, etc.



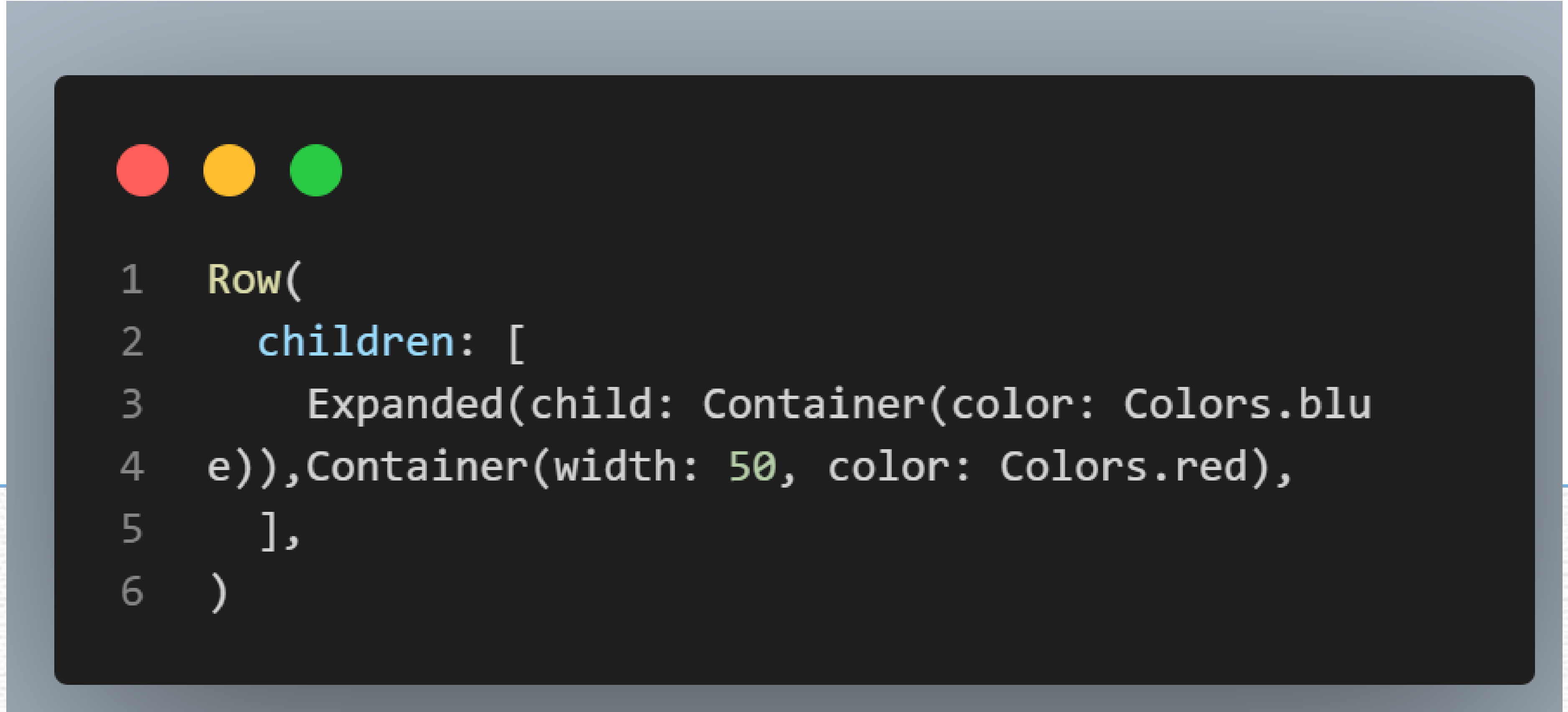
```
var screenWidth = MediaQuery.of(context).size.width;
var screenHeight = MediaQuery.of(context).size.height;
```

Responsive Widgets

2-Flexible Layout Widgets

1. Expanded:

Fills remaining space in Row/Column

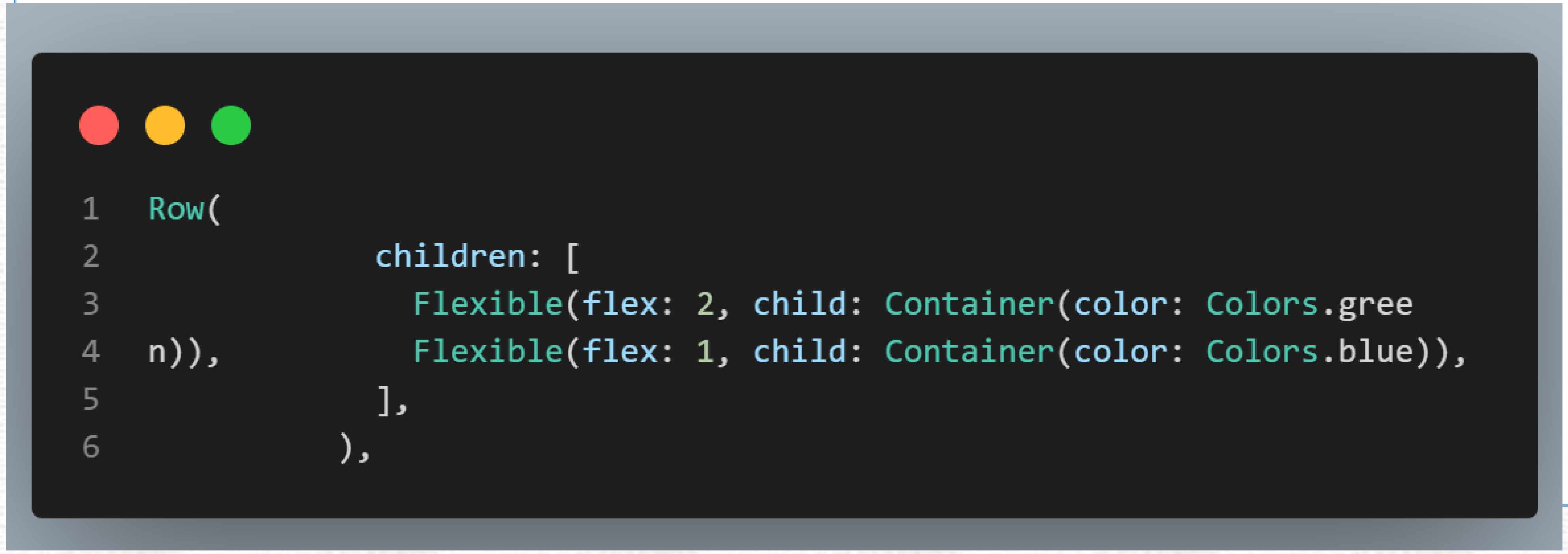


```
Row(
  children: [
    Expanded(child: Container(color: Colors.blu
e)),Container(width: 50, color: Colors.red),
  ],
)
```

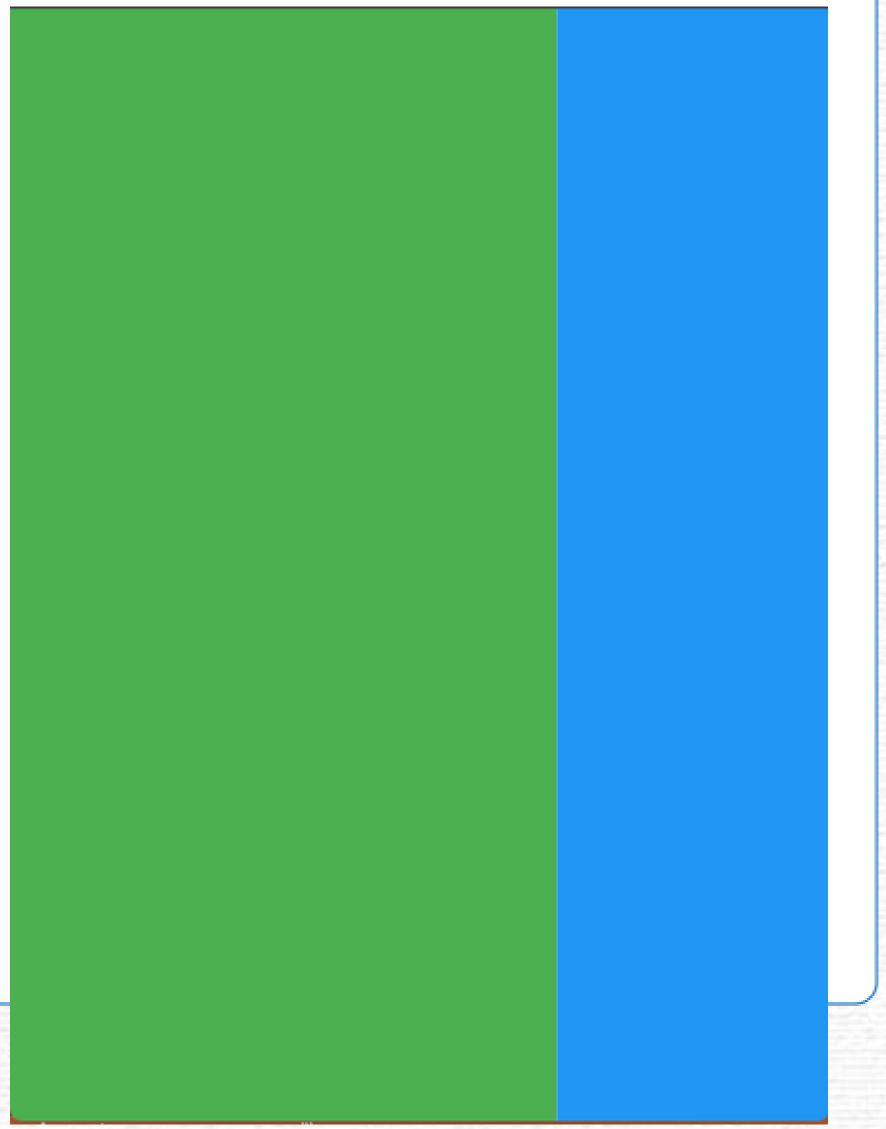
Responsive Widgets

2. Flexible:

Like Expanded but can shrink if needed



```
Row(
      children: [
       Flexible(flex: 2, child: Container(color: Colors.gree
n)),   Flexible(flex: 1, child: Container(color: Colors.blue)),
      ],
     ),
```



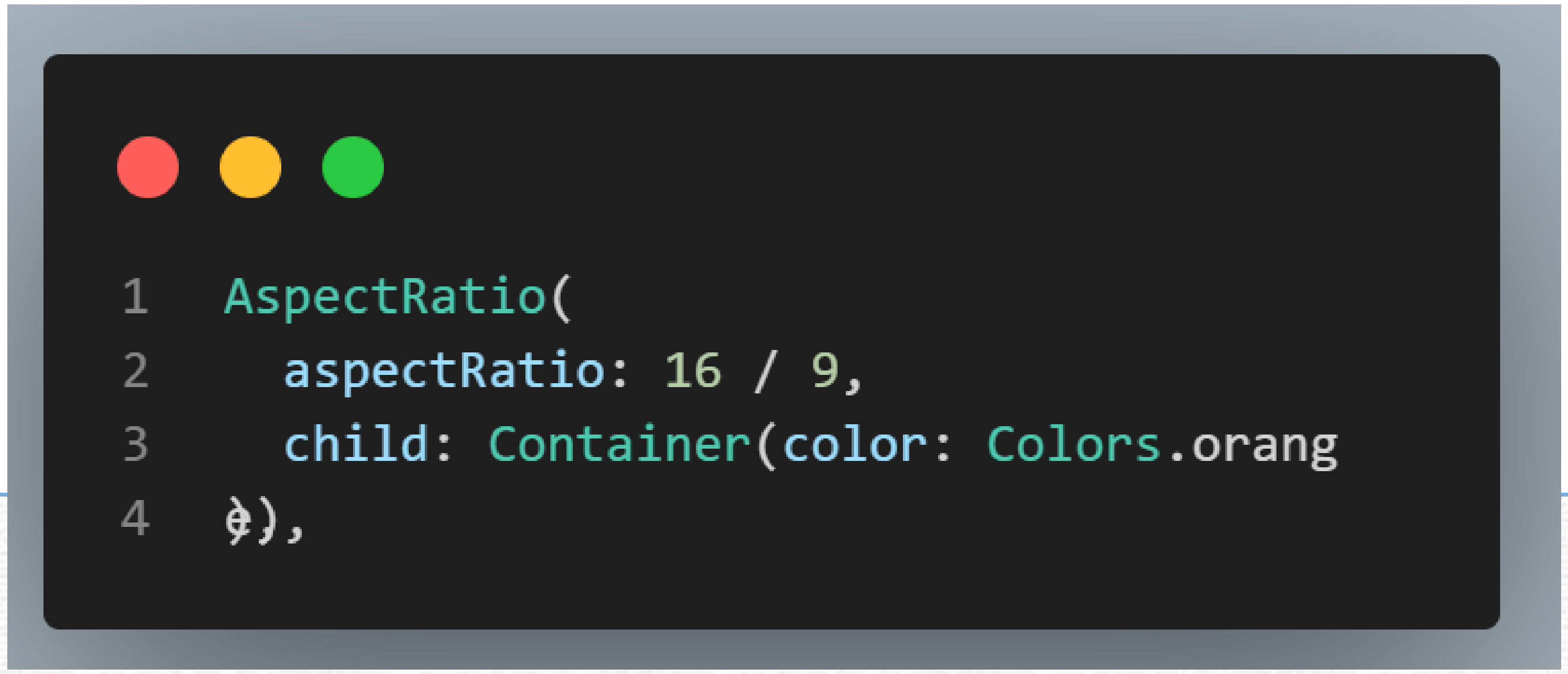
FCI - Helwan
Student Branch

Responsive Widgets

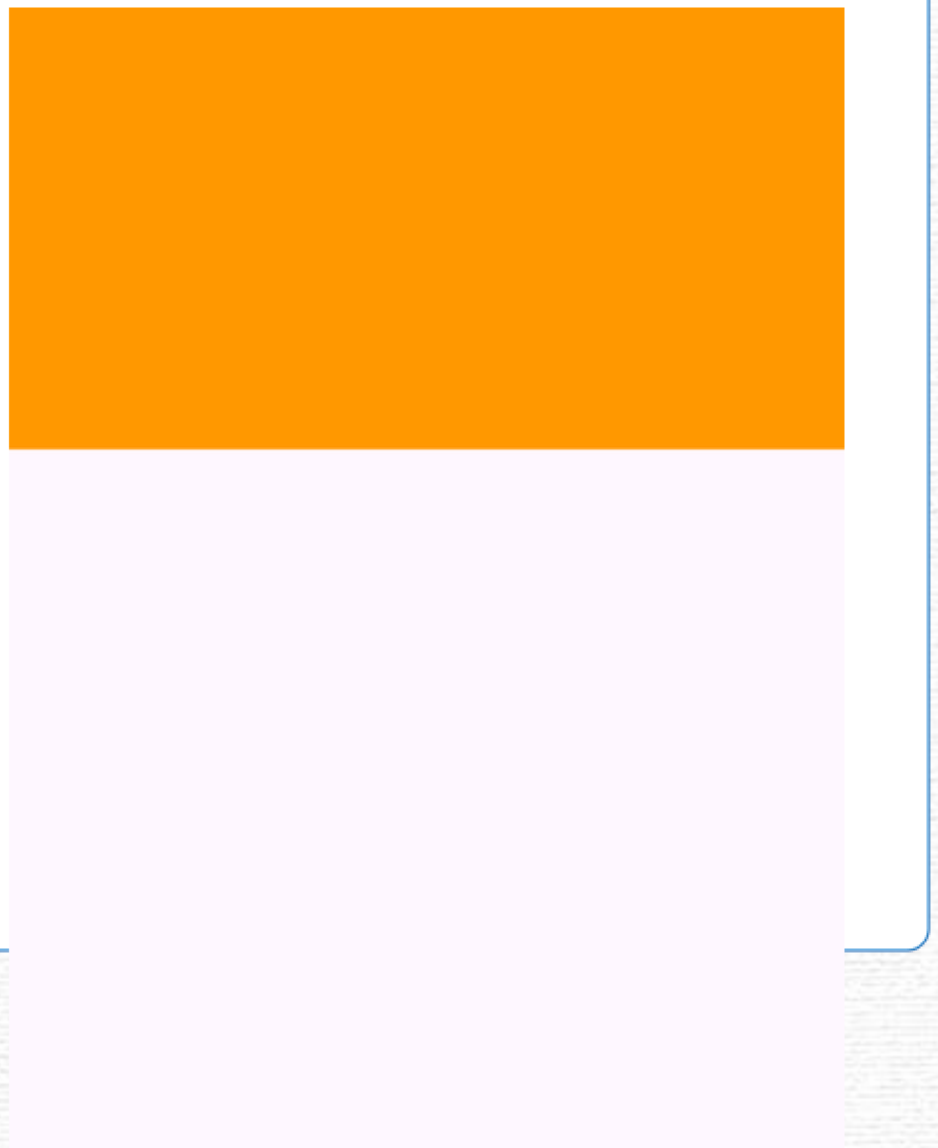
AspectRatio & Intrinsic Dimensions

1. AspectRatio Widget:

Maintains a specific width-to-height ratio



```
AspectRatio(
  aspectRatio: 16 / 9,
  child: Container(color: Colors.orang
e)),
```

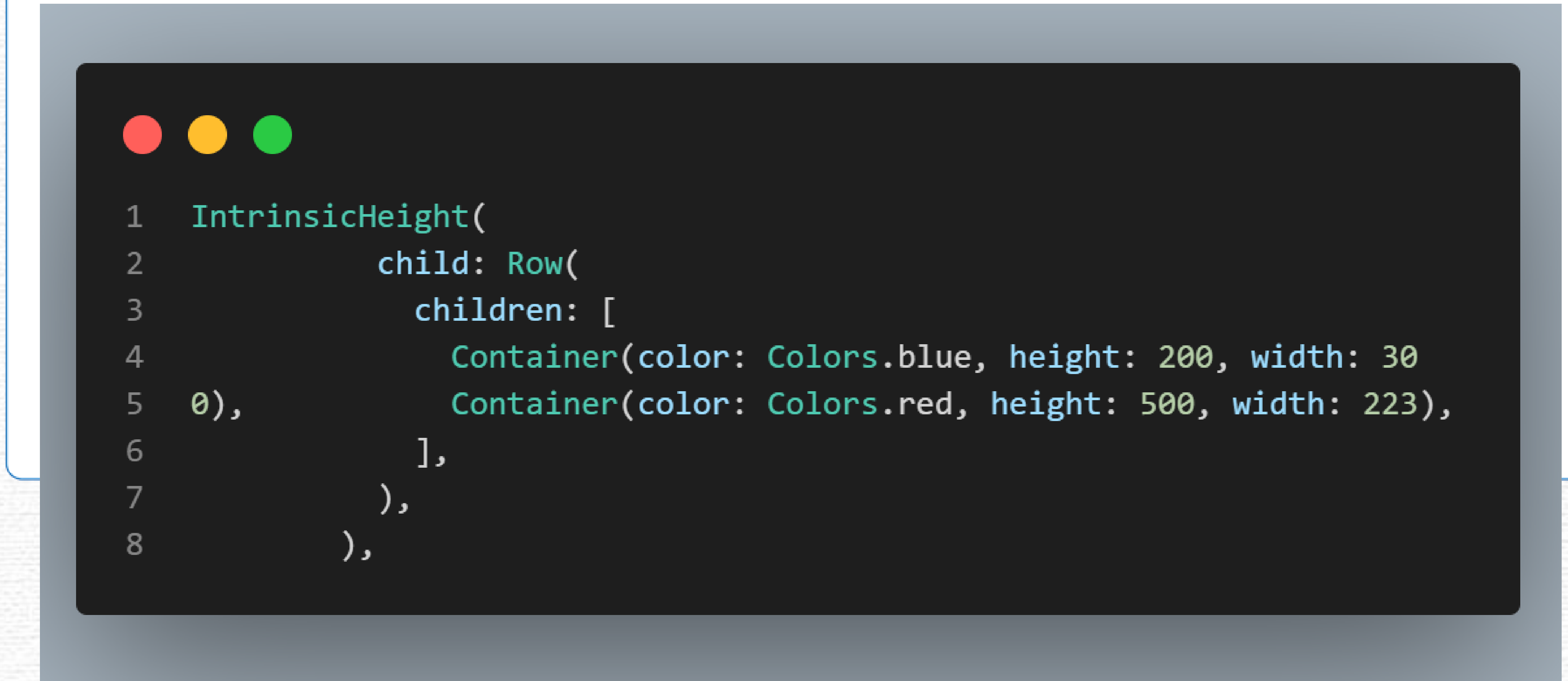


S25' Training Sessions

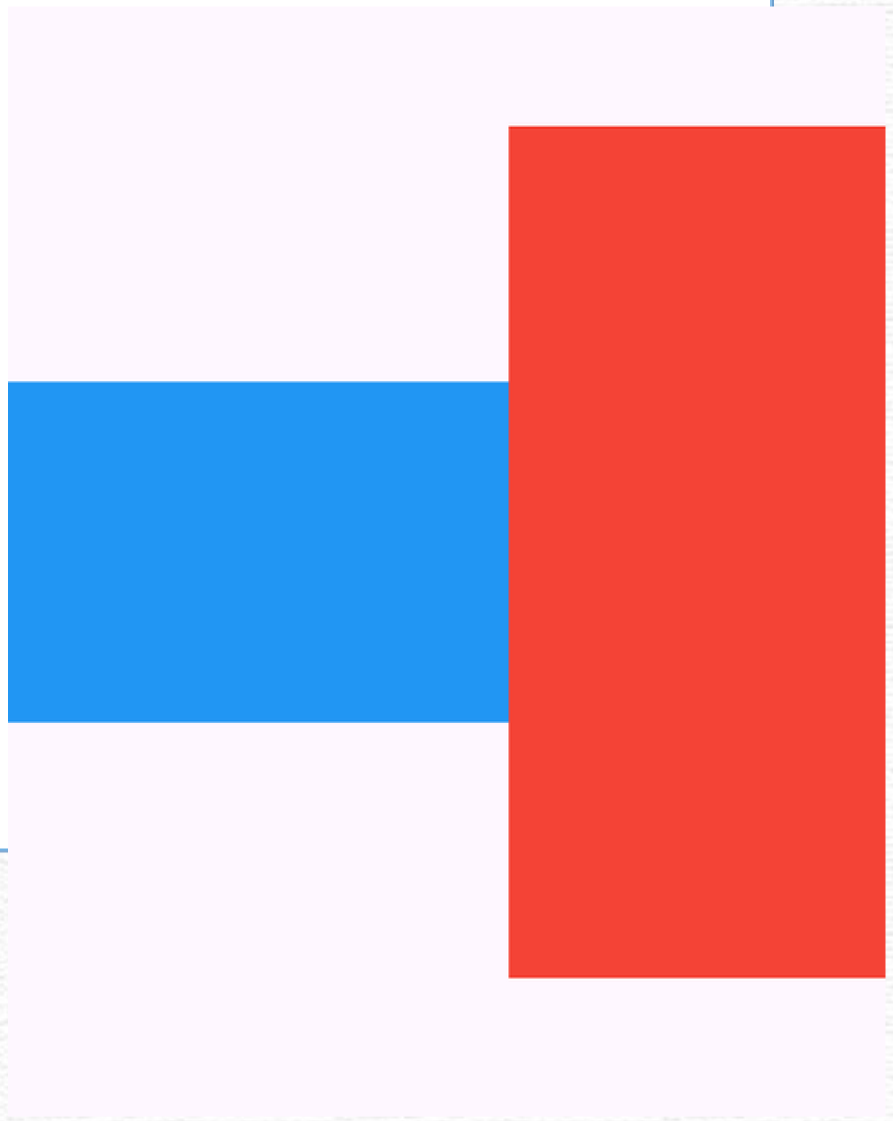
Responsive Widgets

IntrinsicWidth / IntrinsicHeight

Sizes widgets based on their content



```
IntrinsicHeight(
          child: Row(
            children: [
              Container(color: Colors.blue, height: 200, width: 30
0),            Container(color: Colors.red, height: 500, width: 223),
            ],
          ),
        ),
```



Task

Upgrade your Notes App

THANK YOU

SEE YOU NEXT TIME