

앱 개발 입문

성균관대학교 컬처애편테크놀로지융합전공 하계 부트캠프

3일차 - 2023. 06. 26 (월)

강의 슬라이드 링크

https://github.com/kunny/skku-bootcamp-2023-summer/blob/main/_slides/day3.pdf

머티리얼 디자인 알아보기



머티리얼 디자인(Material Design)이란?

- 구글에서 제공하는 디자인 프레임워크로, 다양한 플랫폼 (휴대폰, 태블릿, PC 등)에서 최소한의 수고로도 미려한 디자인의 앱을 제작할 수 있습니다.
- 다양한 플랫폼에서 사용할 수 있는 머티리얼 디자인 컴포넌트를 제공합니다.
- 머티리얼 디자인에서 사용할 수 있는 [아이콘](#) 및 [색상](#)을 지원합니다.
- <https://material.io/>

테스트용 DartPad 구성하기

테스트용 DartPad 구성하기

- DartPad (<https://dartpad.dev>)에 접속합니다.
- 아래 내용을 코드 창에 붙여넣습니다.

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Column Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ),
      home: null,
    );
  }
}
```

테스트용 DartPad 구성하기

- DartPad (<https://dartpad.dev>)에 접속합니다.
- 아래 내용을 코드 창에 붙여넣습니다.


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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Column Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ),
      home: null,
    );
  }
}
```

← 각 위젯을 테스트 할 때마다 수정하게 될
부분

테스트용 DartPad 구성하기

 DartPad

<> New Pad ↺ Reset ≡ Format ⬇ Install SDK

```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return MaterialApp(
9       title: 'Column Demo',
10      theme: ThemeData(
11        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
12        useMaterial3: true,
13      ),
14      home: ColumnDemo(),
15    );
16  }
17 }
18
```

▶ Run

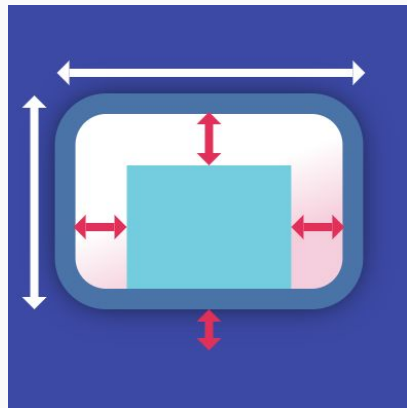
코드를 자동으로
정렬해줍니다.

컨테이너 위젯 알아보기



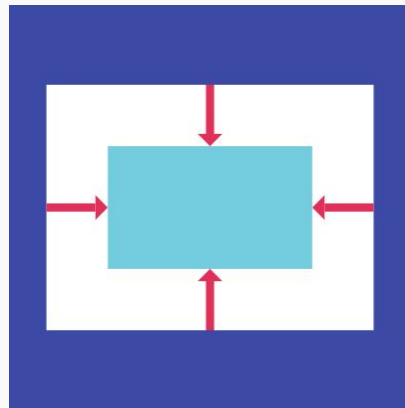
Container

- 위젯을 배치할 수 있는 다양한 방법을 지원합니다.
- 주요 속성
 - child: 자식 위젯을 지정합니다.
 - width: 컨테이너 너비를 지정합니다.
 - height: 컨테이너 높이를 지정합니다.
 - color: 컨테이너 배경 색상을 설정합니다.
 - padding: 컨테이너 내부 여백을 설정합니다.
 - margin: 컨테이너 외부 여백을 설정합니다.
 - alignment: 자식 위젯의 정렬 방법을 결정합니다.



Center

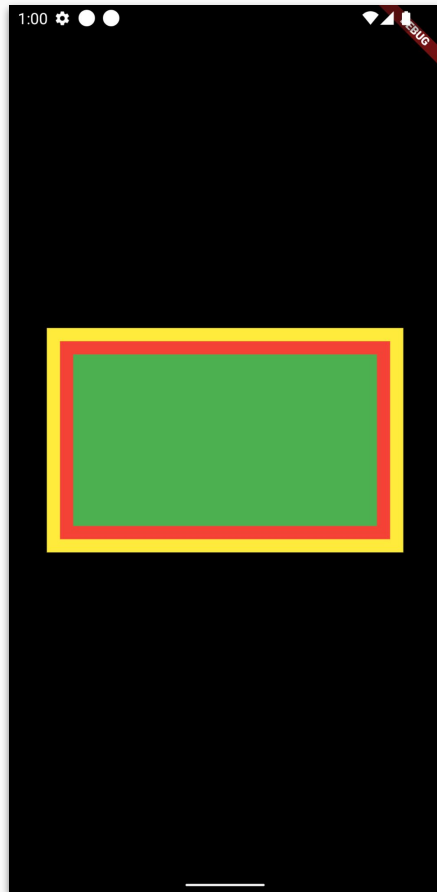
- 자식 위젯을 컨테이너의 정가운데에 배치합니다.
- 주요 속성
 - child: 자식 위젯을 지정합니다.



Demo

```
class ContainerDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Center(  
      child: Container(  
        color: Colors.yellow,  
        child: Container(  
          color: Colors.red,  
          width: 300,  
          height: 180,  
          margin: const EdgeInsets.all(12.0),  
          padding: const EdgeInsets.all(12.0),  
          child: Container(color: Colors.green),  
        ),  
      ),  
    );  
  }  
}
```

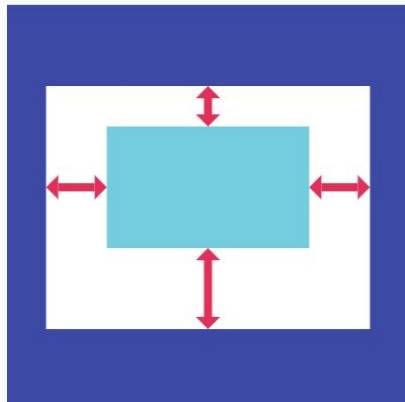
```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      ...  
      home: ContainerDemo(),  
    );  
  }  
}
```





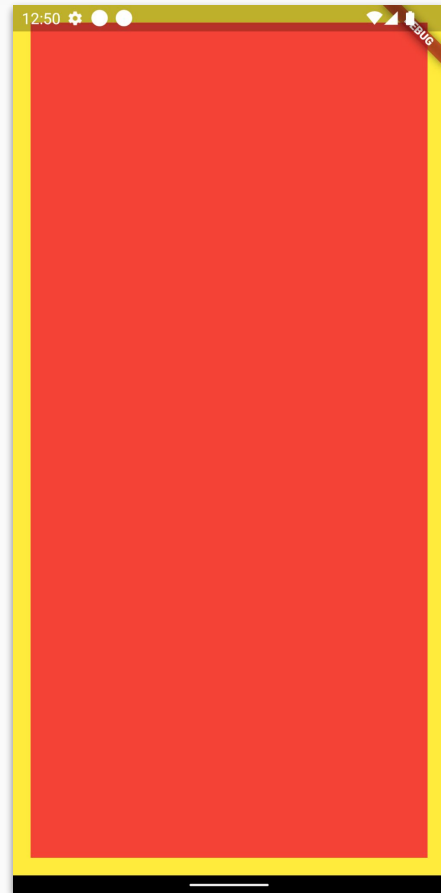
Padding

- 컨테이너 위젯 내부에 여백을 추가합니다.
- 주요 속성
 - child: 자식 위젯을 지정합니다.
 - padding: 여백을 넣어줄 위치 및 값을 지정합니다.



Demo

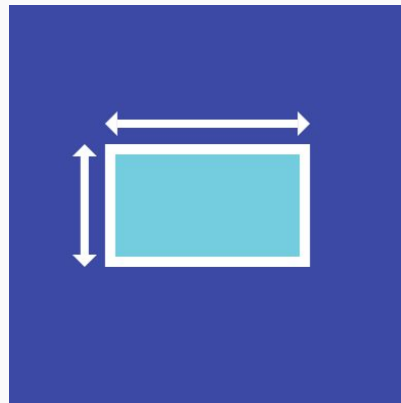
```
class PaddingDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Container(  
      color: Colors.yellow,  
      child: Padding(  
        padding: const EdgeInsets.all(16.0),  
        child: Container(color: Colors.red),  
      ),  
    );  
  }  
}
```





SizedBox

- 너비와 높이를 지정할 수 있는 컨테이너입니다.
- 주요 속성
 - child: 자식 위젯을 지정합니다.
 - width: 컨테이너 너비를 지정합니다.
 - height: 컨테이너 높이를 지정합니다.

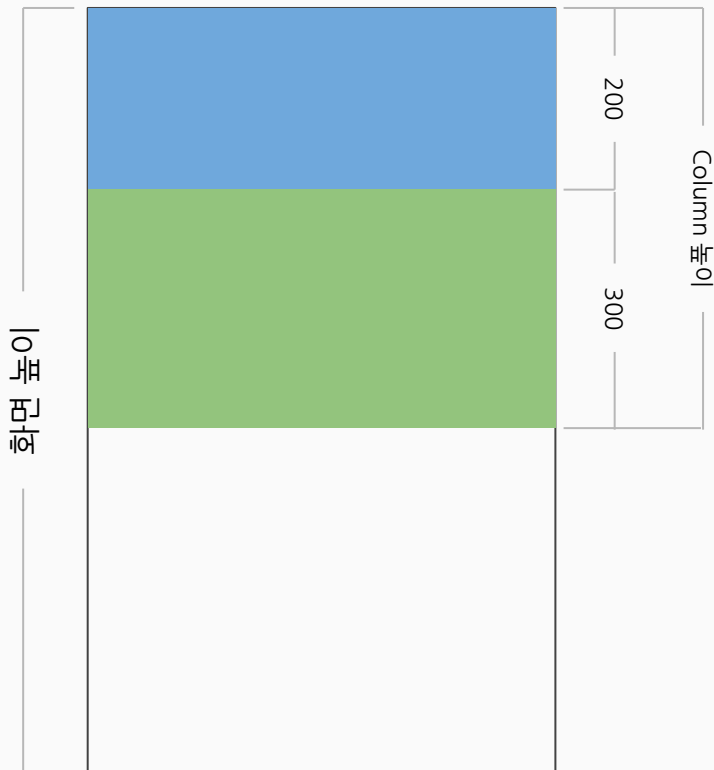


Column

- 자식 위젯들을 수직 방향으로 배치할 수 있는 컨테이너입니다.
- 주요 속성
 - children: 자식 위젯을 지정합니다.
 - mainAxisAlignment: 수직 방향 정렬 방법을 결정합니다.
 - mainAxisSize: 수직 방향으로 공간을 얼마나 차지할지 결정합니다.
 - crossAxisAlignment: 자식 위젯들을 수평 방향으로 어떻게 정렬할지 결정합니다.



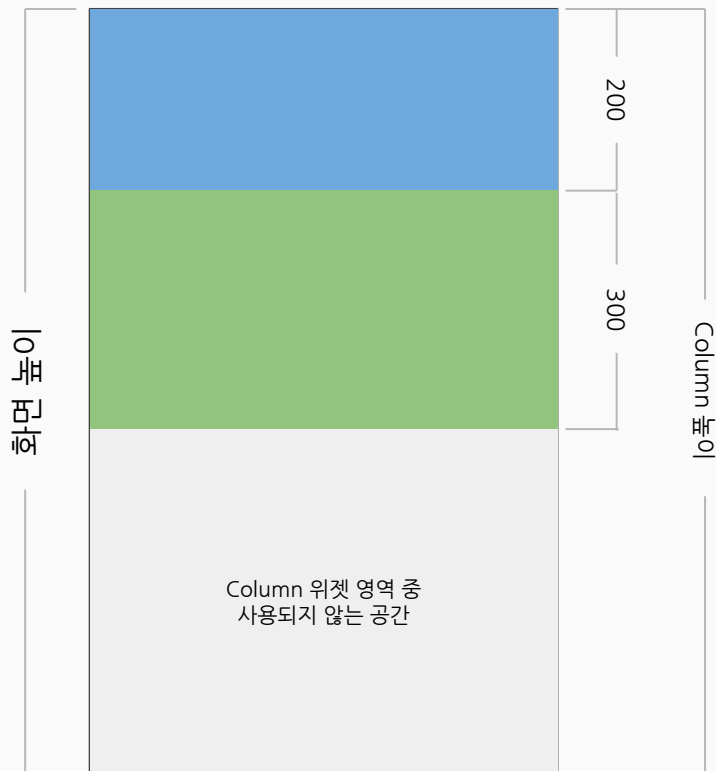
Column: MainAxisSize.min



```
Column(  
  mainAxisAlignment: MainAxisAlignment.min,  
  children: <Widget>[  
    Container(  
      height: 200,  
      color: Colors.blue,  
    ),  
    Container(  
      height: 300,  
      color: Colors.green,  
    ),  
  ],  
)
```

Column 위젯에 포함된 위젯이 차지하는 만큼만 공간을 차지합니다.

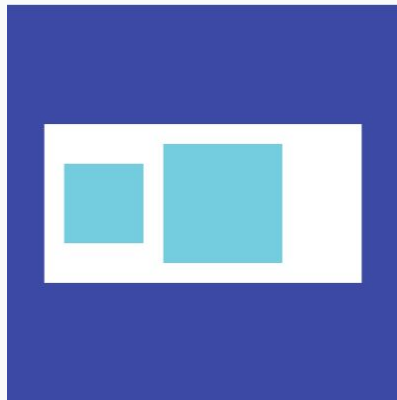
Column: MainAxisSize.max



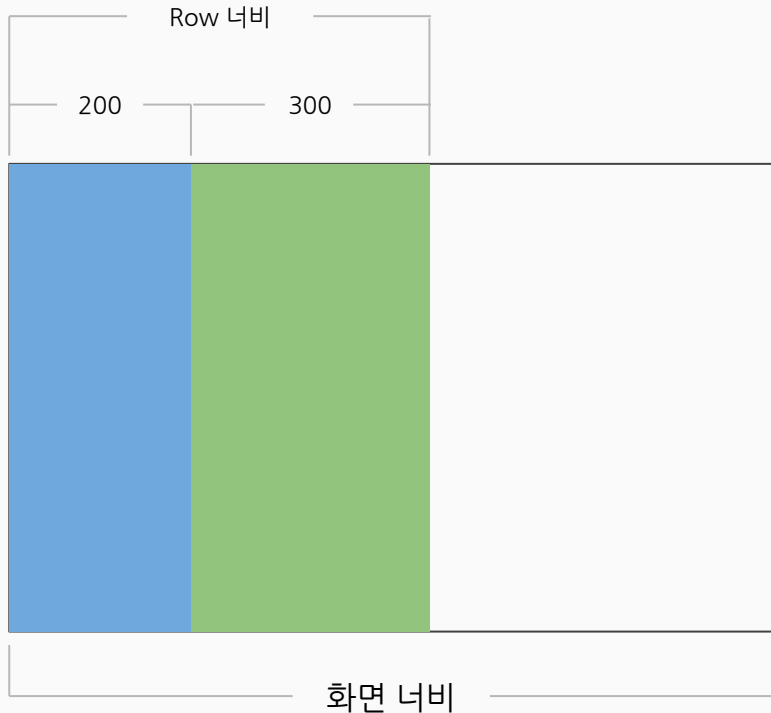
```
Column(  
  mainAxisAlignment: MainAxisAlignment.max,  
  children: <Widget>[  
    Container(  
      height: 200,  
      color: Colors.blue,  
    ),  
    Container(  
      height: 300,  
      color: Colors.green,  
    ),  
  ],  
)
```

Column 위젯이 차지할 수 있는 최대
높이만큼 공간을 차지합니다.

- 자식 위젯들을 수평 방향으로 배치할 수 있는 컨테이너입니다.
- 주요 속성
 - children: 자식 위젯을 지정합니다.
 - mainAxisAlignment: 수평 방향 정렬 방법을 결정합니다.
 - mainAxisSize: 수평 방향으로 공간을 얼마나 차지할지 결정합니다.
 - crossAxisAlignment: 자식 위젯들을 수직 방향으로 어떻게 정렬할지 결정합니다.



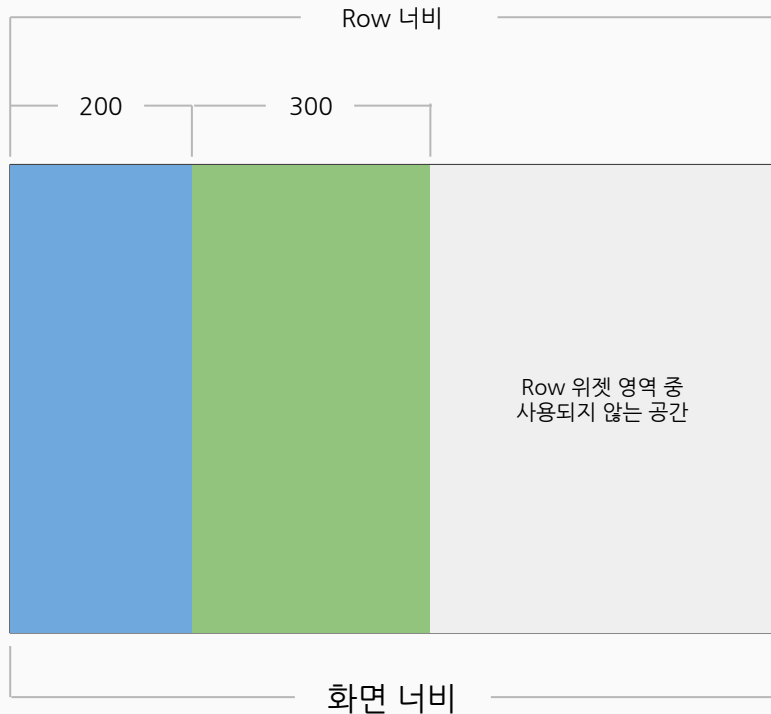
Row: MainAxisSize.min



```
Row(  
  mainAxisAlignment: MainAxisSize.min,  
  children: <Widget>[  
    Container(  
      width: 200,  
      color: Colors.blue,  
    ),  
    Container(  
      width: 300,  
      color: Colors.green,  
    ),  
  ],  
)
```

Row 위젯에 포함된 위젯이 차지하는 만큼만 공간을 차지합니다.

Row: MainAxisSize.max



```
Row(  
  mainAxisSize: MainAxisSize.max,  
  children: <Widget>[  
    Container(  
      width: 200,  
      color: Colors.blue,  
    ),  
    Container(  
      width: 300,  
      color: Colors.green,  
    ),  
  ],  
)
```

Row 위젯이 차지할 수 있는
최대 너비만큼 공간을 차지합니다.

Expanded

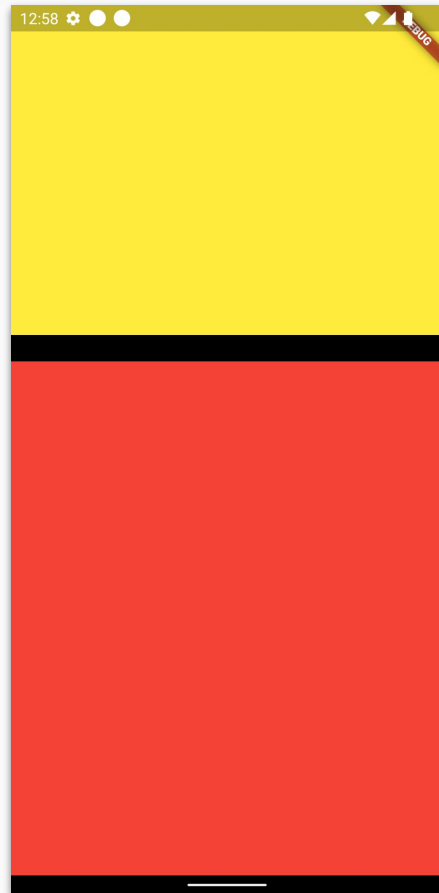
- Column/Row 위젯 내부의 남은 공간을 모두 차지해야 할 때 사용합니다.
- 주요 속성
 - child: 자식 위젯을 지정합니다.

SingleChildScrollView

- 위젯이 화면 크기보다 더 많은 공간을 차지하는 경우, 스크롤을 통해 나머지 부분을 볼 수 있게 해 줍니다.
- 주요 속성
 - child: 자식 위젯을 지정합니다.
 - scrollDirection: 스크롤 방향을 지정합니다.

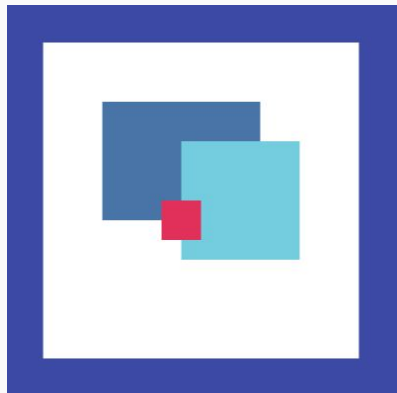
Demo

```
class ColumnDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: SingleChildScrollView(  
        child: SizedBox(  
          height: 2000,  
          child: Column(  
            mainAxisAlignment: MainAxisAlignment.min,  
            children: [  
              Container(height: 300, color: Colors.yellow),  
              const SizedBox(height: 24),  
              Expanded(  
                child: Container(color: Colors.red),  
              ),  
            ],  
          ),  
        ),  
      ),  
    );  
  }  
}
```



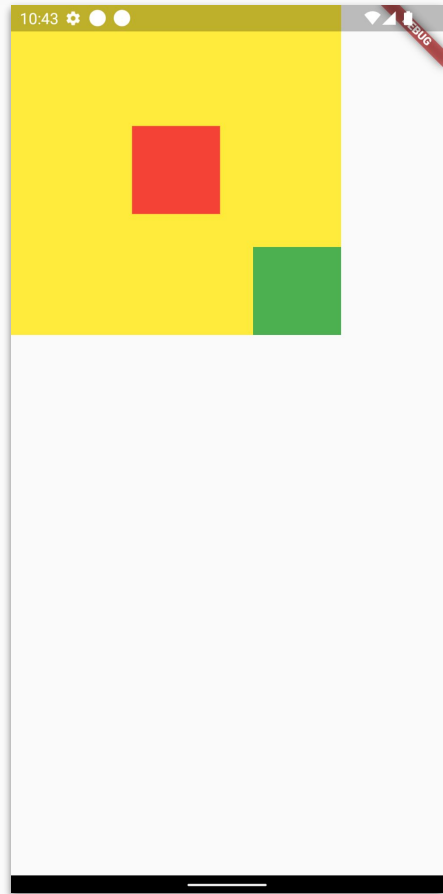


- 자식 위젯들을 포개어 배치할 수 있는 컨테이너입니다. Align 위젯을 함께 사용하면 자식 위젯을 다양한 방법으로 배치할 수 있습니다.
- 주요 속성
 - children: 자식 위젯을 지정합니다.
 - alignment: 별도로 정렬 방법이 정의되지 않은 자식 위젯을 어떻게 정렬할지 결정합니다.



Demo

```
class StackDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: Container(  
        width: 300,  
        height: 300,  
        color: Colors.yellow,  
        child: Stack(  
          children: [  
            Align(  
              alignment: Alignment.center,  
              child: Container(  
                width: 80,  
                height: 80,  
                color: Colors.red,  
              ),  
            ),  
            Align(  
              alignment: Alignment.bottomRight,  
              child: Container(  
                width: 80,  
                height: 80,  
                color: Colors.green,  
              ),  
            ),  
          ],  
        ),  
      ),  
    );  
  }  
}
```



머티리얼 디자인 컴포넌트로 화면 구성하기

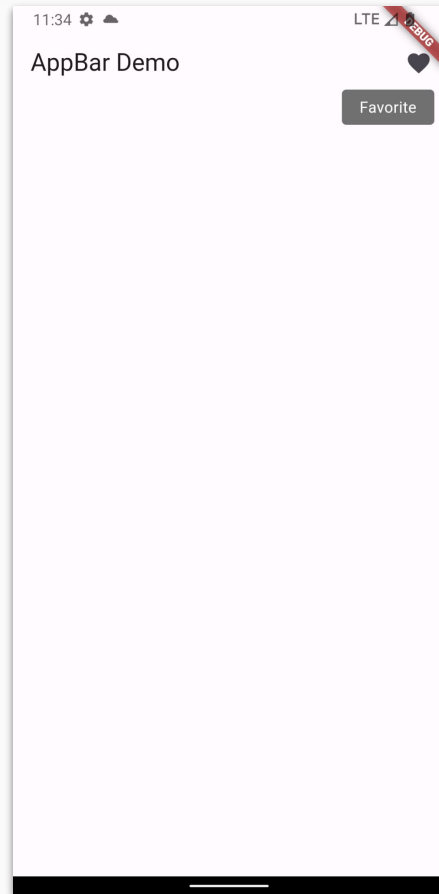
Flutter Gallery
(<https://gallery.flutter.dev>)

직접 실습해볼 위젯들

1. AppBar, IconButton, CircularProgressIndicator
2. Buttons (Text, Elevated, Outlined, FloatingActionButton)
3. ListTile, CircleAvatar
4. TextField

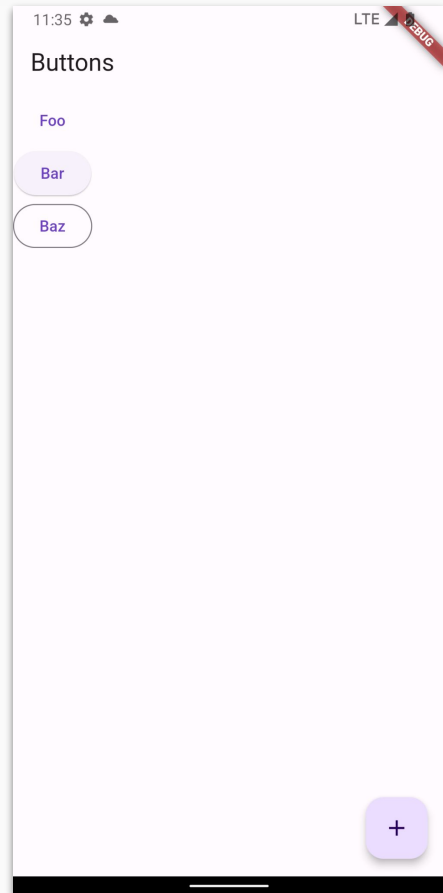
AppBar, IconButton, CircularProgressIndicator

```
class AppBarDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text('AppBar Demo'),  
        actions: [  
          IconButton(  
            icon: const Icon(Icons.favorite),  
            tooltip: 'Favorite',  
            onPressed: () {  
              print('Clicked Favorite');  
            },  
          ),  
        ],  
      ),  
    );  
  }  
}
```



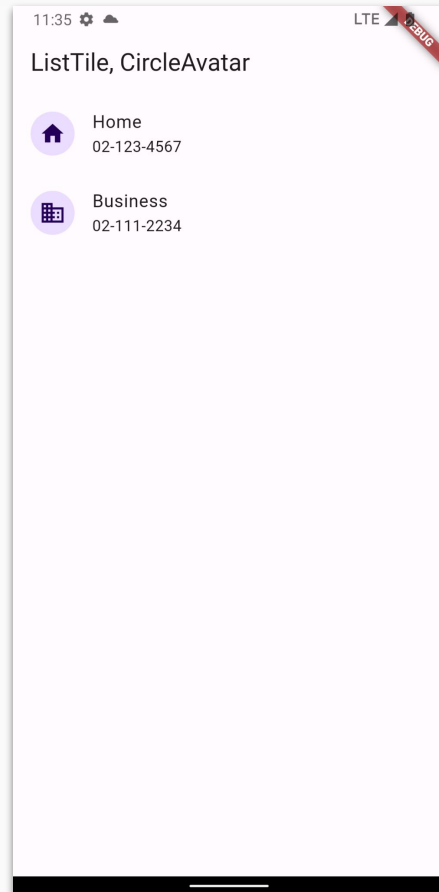
Buttons (Text, Elevated, Outlined, FloatingActionButton)

```
class ButtonDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text('Buttons'),  
      ),  
      body: Column(  
        children: [  
          TextButton(  
            onPressed: () {},  
            child: const Text('Foo'),  
          ),  
          ElevatedButton(  
            onPressed: () {},  
            child: const Text('Bar'),  
          ),  
          OutlinedButton(  
            onPressed: () {},  
            child: const Text('Baz'),  
          ),  
        ],  
      ),  
      floatingActionButton: FloatingActionButton(  
        onPressed: () {},  
        tooltip: 'Add',  
        child: const Icon(Icons.add),  
      ),  
    );  
  }  
}
```



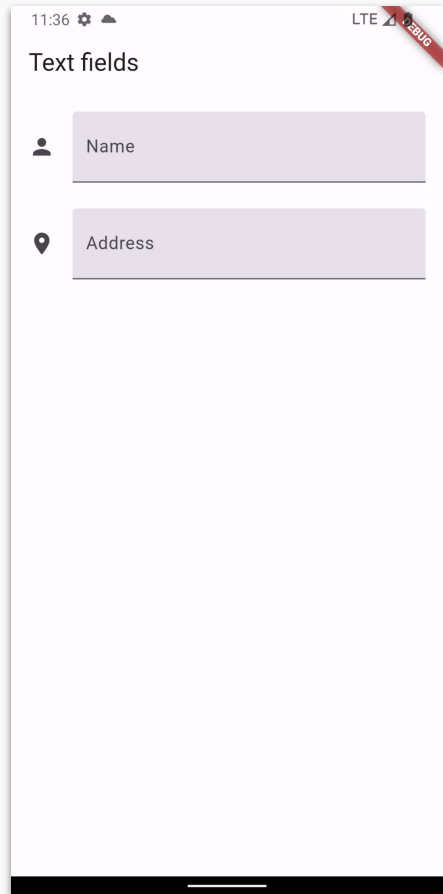
ListTile, CircleAvatar

```
class ListTileDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text('ListTile, CircleAvatar'),  
      ),  
      body: Column(  
        children: [  
          ListTile(  
            leading: const CircleAvatar(  
              child: Icon(Icons.home),  
            ),  
            title: const Text('Home'),  
            subtitle: const Text('02-123-4567'),  
            onTap: () {},  
          ),  
          ListTile(  
            leading: const CircleAvatar(  
              child: Icon(Icons.business),  
            ),  
            title: const Text('Business'),  
            subtitle: const Text('02-111-2234'),  
            onTap: () {},  
          ),  
        ],  
      ),  
    );  
  }  
}
```



TextField

```
class TextFieldDemo extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text('Text fields'),  
      ),  
      body: Padding(  
        padding: const EdgeInsets.all(16.0),  
        child: Column(  
          children: const [  
            TextField(  
              decoration: InputDecoration(  
                icon: Icon(Icons.person),  
                labelText: 'Name',  
                filled: true,  
              ),  
            ),  
            SizedBox(height: 24.0),  
            TextField(  
              decoration: InputDecoration(  
                icon: Icon(Icons.place),  
                labelText: 'Address',  
                filled: true,  
              ),  
            ),  
          ],  
        ),  
      ),  
    );  
  }  
}
```



노트 앱 프로젝트 만들기

- Project name: `sticky_notes`
- Organization: `edu.skku.sco`. [본인 닉네임 혹은 ID]
 - 예) 닉네임이 androidhuman인 경우: `edu.skku.sco.androidhuman`
- Platforms: `Android, iOS`

- `pubspec.yaml` 내 주석 삭제
- `test` 폴더 삭제
- 노트 목록 화면 뼈대 작성
- 앱 첫 화면 변경 및 앱 테마 색상 변경

pubspec.yaml 내 주석 삭제

```
name: sticky_notes
description: A new Flutter project.
publish_to: 'none'
version: 1.0.0+1

environment:
  sdk: '>=3.0.5 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true
```

pubspec.yaml 내 주석 삭제

```
name: sticky_notes
description: A new Flutter project.
publish_to: 'none'
version: 1.0.0+1

environment:
  sdk: '>=3.0.5 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true
```

pubspec.yaml 파일은 YAML 형식을 사용하며, 이는 공백 문자를 사용하여 하위 설정을 구분합니다. (2 spaces)

공백 문자는 항상 짝수 단위로만 사용하며, 이를 지키지 않으면 오류가 발생하니 유의해야 합니다.

노트 목록 화면 뼈대 작성

1. `lib` 폴더 아래에 `page` 폴더를 생성합니다.
2. `page` 폴더에 `note_list_page.dart` 파일을 생성합니다.
3. 노트 목록을 표시할 위젯인 `NoteListPage` 코드를 작성합니다.

노트 목록 화면 뼈대 작성

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

class NoteListPage extends StatefulWidget {
  const NoteListPage(super.key);

  @override
  State createState() => _NoteListPageState();
}

class _NoteListPageState extends State<NoteListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Sticky Notes'),
      ),
    );
  }
}
```

note_list_page.dart

앱 첫 화면 변경 및 앱 테마 색상 변경

1. **MyApp** 클래스의 **home** 속성에 `NoteListPage`를 지정하여 첫 화면으로 노트 목록 페이지를 표시합니다.
2. **MyApp** 클래스의 **title** 속성을 변경합니다.
3. **MyApp** 클래스의 **theme** 속성에 할당되어있는 **ThemeData.colorScheme** 속성을 변경하여 앱 테마 색상을 원하는 색상으로 변경합니다.
4. **MyHomePage** 및 **_MyHomePageState** 클래스를 제거합니다.

앱 첫 화면 변경 및 앱 테마 색상 변경

```
import 'package:flutter/material.dart';
import 'package:sticky_notes/page/note_list_page.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Sticky Notes',
      theme: ThemeData(
        primarySwatch: Colors.teal,
      ),
      home: const NoteListPage(),
    );
  }
}
```

main.dart

노트 데이터를 저장할 수 있는
클래스 만들기

노트 데이터를 저장할 수 있는 클래스 만들기

1. `lib` 폴더 아래에 `data` 폴더를 생성합니다.
2. `data` 폴더에 `note.dart` 파일을 생성합니다.
3. 노트에 저장할 데이터를 정의한 후, `Note` 클래스를 작성합니다.

노트 데이터를 저장할 수 있는 클래스 만들기

```
class Note {  
  final String title;  
  
  final String body;  
  
  Note(  
    this.body, {  
    this.title = '',  
  });  
}
```

노트 데이터를 저장할 수 있는 클래스 만들기

```
Note(  
    this.body, {  
        this.title = '',  
    });
```

생성자: 클래스의 새로운 인스턴스를 만들때
사용합니다.

```
// body = '샘플 노트입니다', title = '샘플 노트'  
Note('샘플 노트입니다', title: '샘플 노트');
```

```
// body = '제목 없는 노트', title = ''  
Note('제목 없는 노트');
```

완성된 코드

https://github.com/kunny/skku-bootcamp-2023-summer/tree/main/sticky_notes/step1