



# Flutter

## 3일차



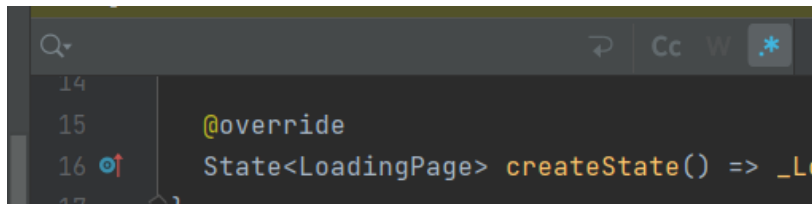
DGROID  
LEE HAIM



# (인생이 편해지는) 플러터 사용법

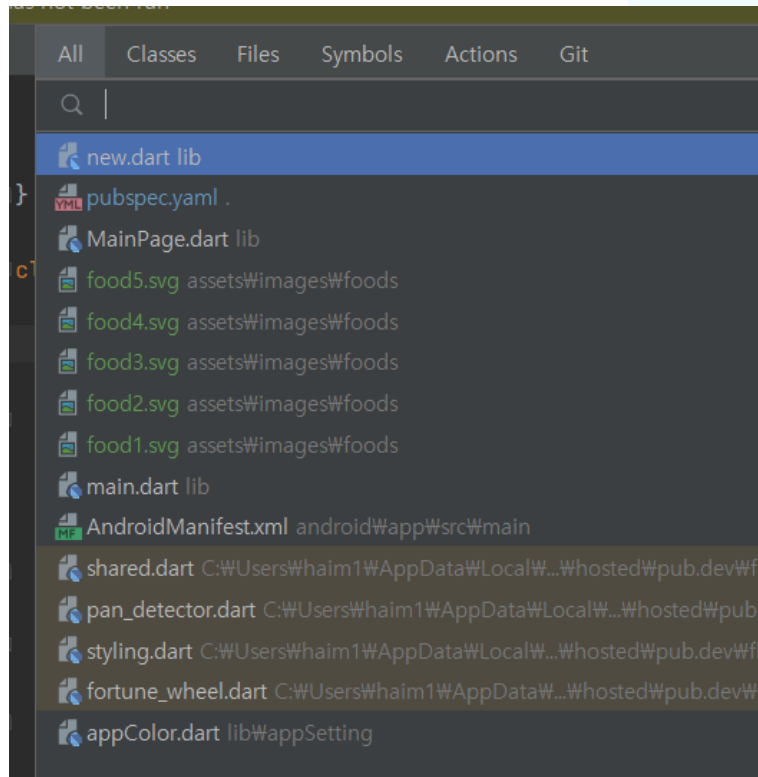
## Ctrl + F

이 파일 내에서 코드 검색하기



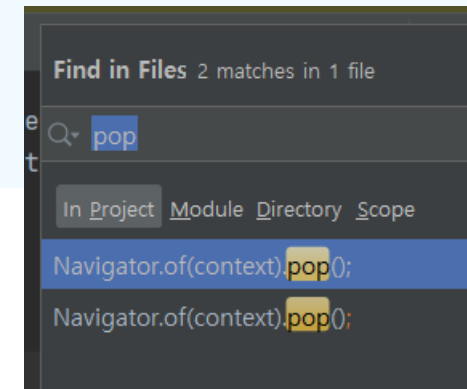
## shift , shift

전체 파일 내에서 파일 검색하기



## Ctrl + Shift + F

전체 파일 내에서 코드 검색하기

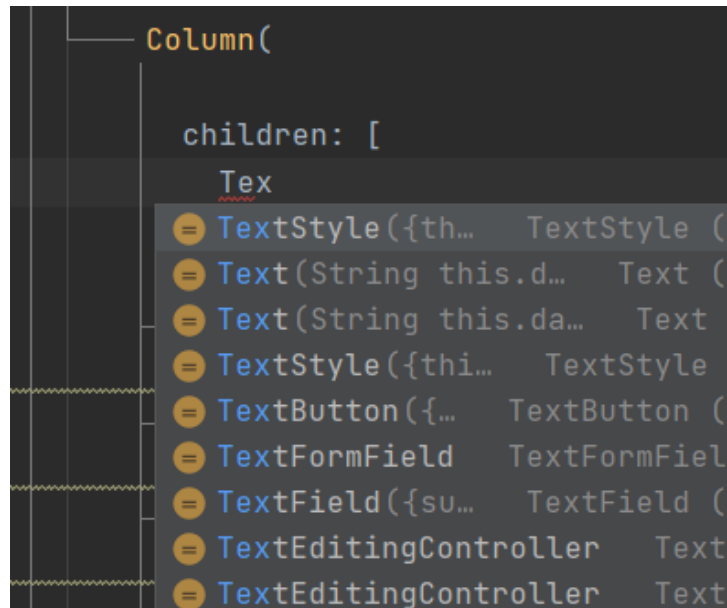




# (인생이 편해지는) 플러터 사용법

## 제발 자동완성 쓰기 !!

오타를 획기적으로 줄여준다.  
필수 매개변수를 쉽게 알 수 있다.  
코딩 시간이 단축된다.



맨 앞 글자를 적으면 관련 Class 등이 뜬.  
맨 첫글자는 대소문자를 구별해주는 것이 좋음.

예상하는 글자를 적어 검색과 비슷한 기능을 기대해볼 수 있음.

거의 대부분의 매개변수, Class, 내가 정의한 변수 등을 자동 완성해줌.



# Layout

3일차 Flutter



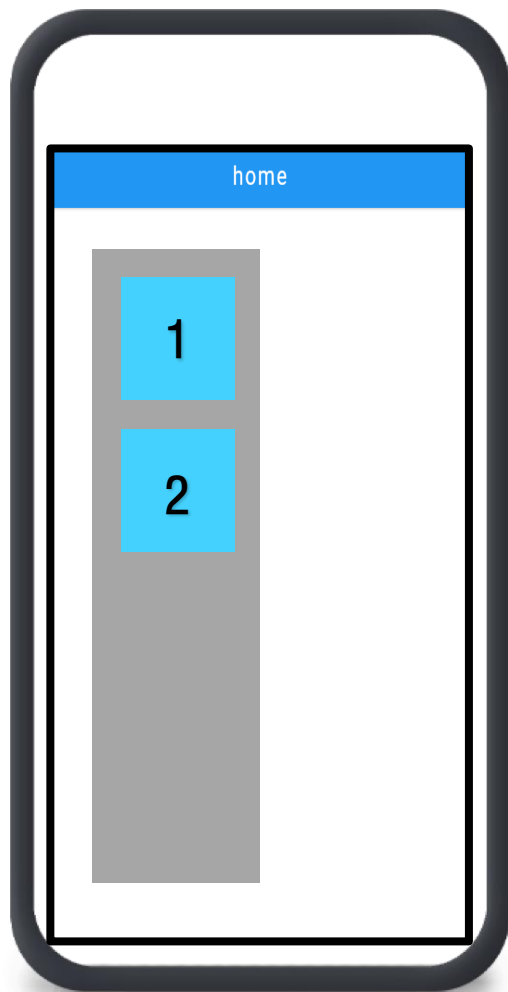
DGROID  
LEE HAIM



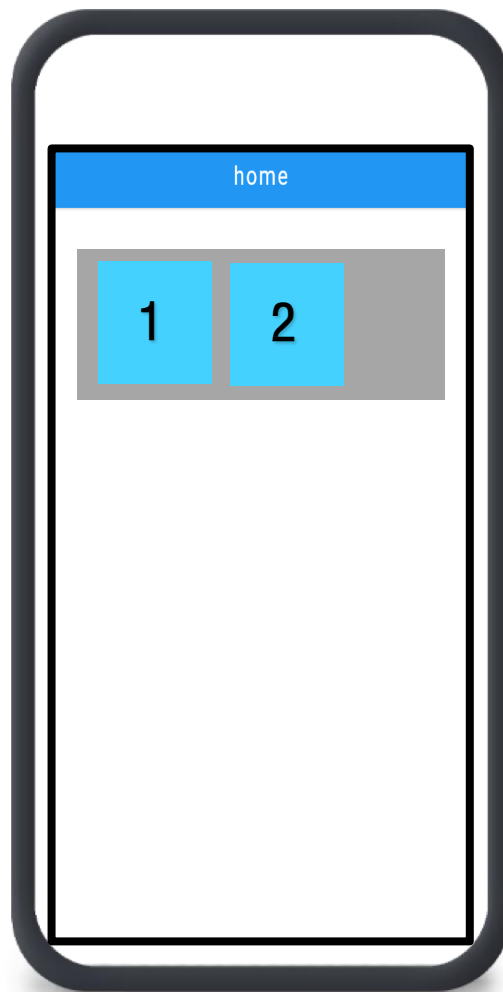
# Flutter 구조

(저번주에 했던 내용)

\*



Column



Row

Body 내부에 레이아웃을 형성할 수 있다.

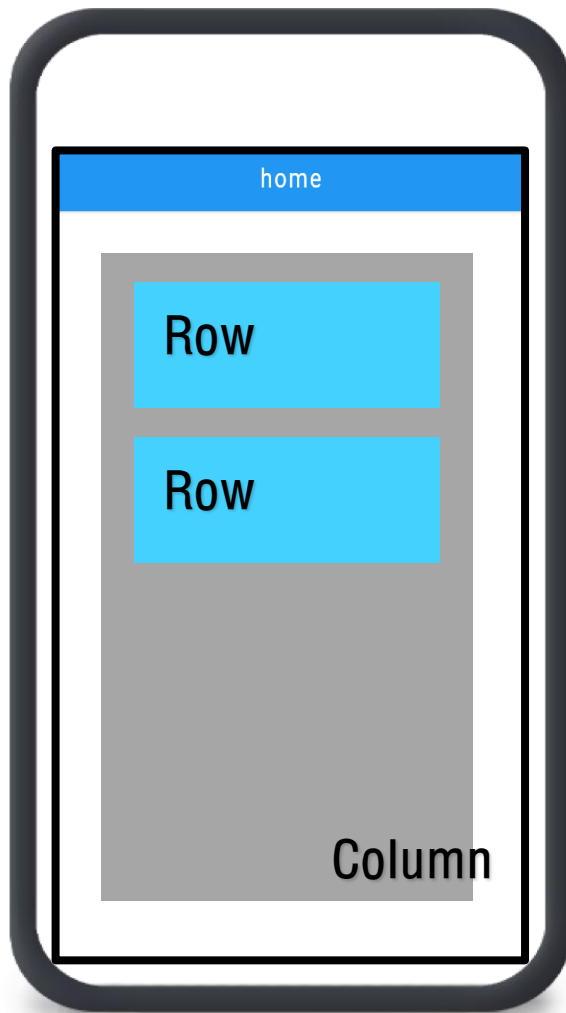
Column은 세로  
Row는 가로로 형성되는 레이아웃이다.

각각의 레이아웃은 왼쪽 그림과 같이  
코드의 순서에 따라 들어가는 위치 순서가 달라진다.

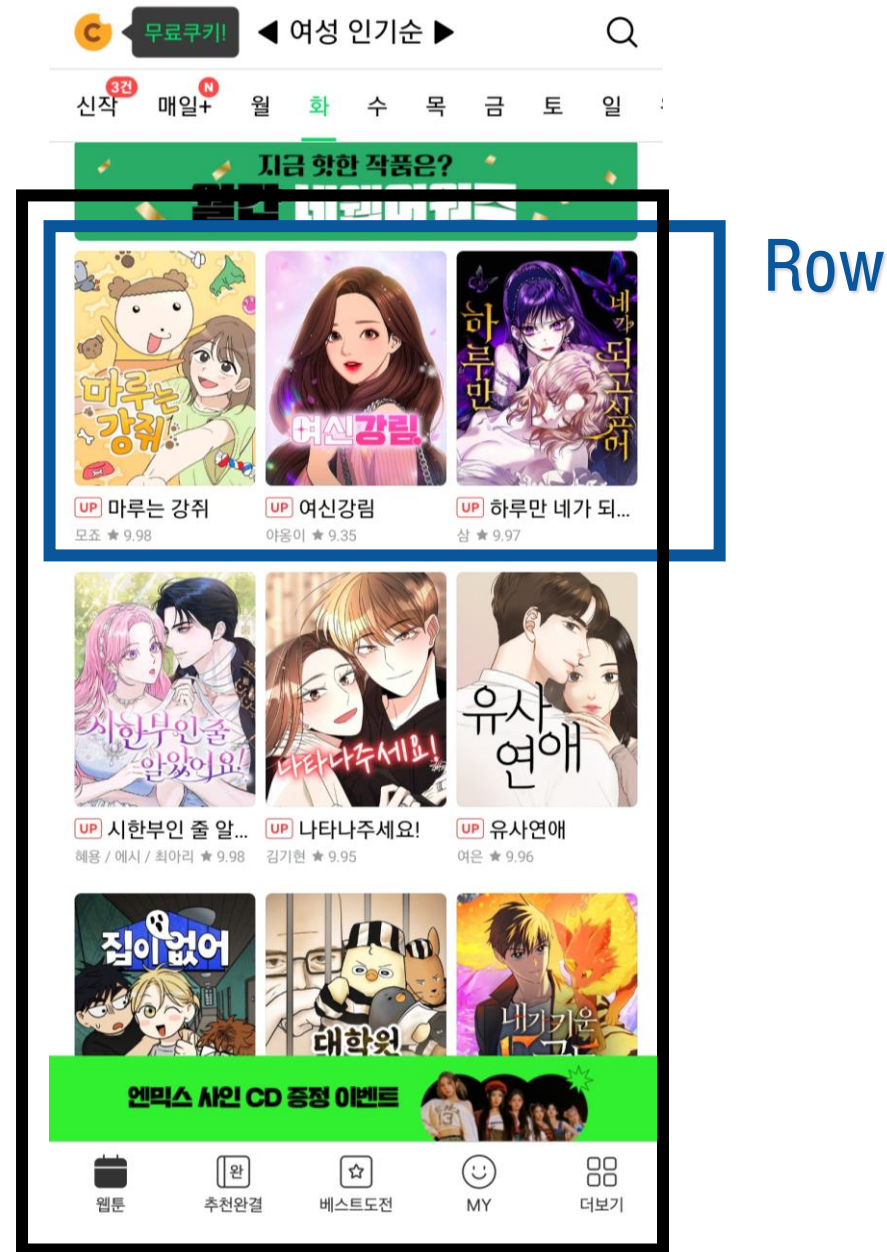
\* 레이아웃 : 이미지, 텍스트 등이 들어갈 방과 같은 개념

# Flutter 구조

복잡한 레이아웃을 구성할 수 있다.



\* children 안에 요소들을 넣는다.



Column

Row



# Pages

3일차 Flutter



DGROID  
LEE HAIM



# StateLess & StateFul

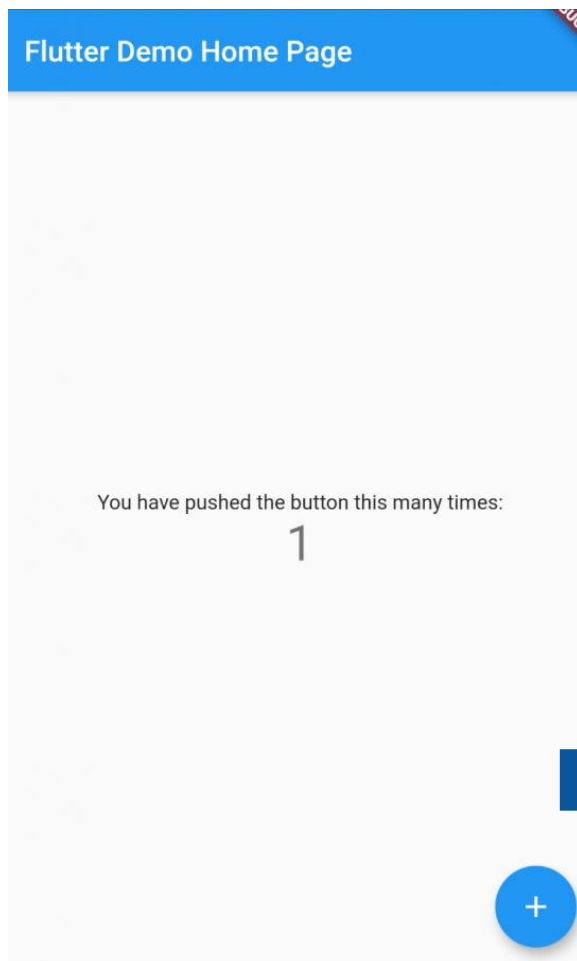
## State Less

움직임이 없는 페이지

Hot Reload 실행 이외에  
화면이 바뀌지 않는다.

(실제로 앱을 깔면  
화면이 바뀌지 않는다.)

- 페이지의 push pop과는  
상관없음.



setState()

```
void _incrementCounter() {  
  setState(() {  
    _counter++;  
  });  
}
```

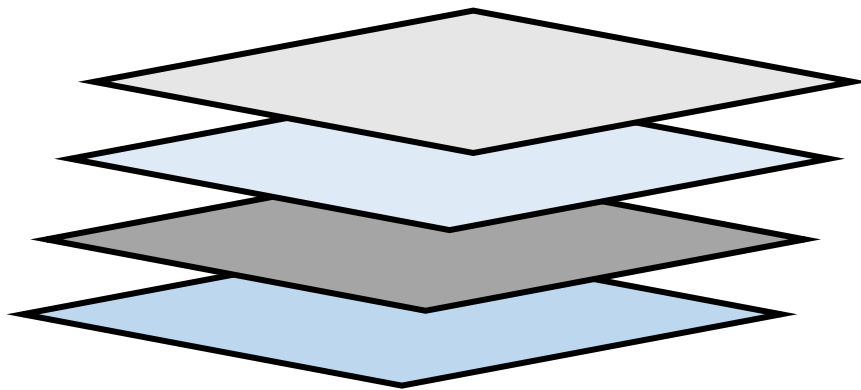
화면을 바꾸도록 해주는 함수.

## State Ful

움직임이 필요한 페이지

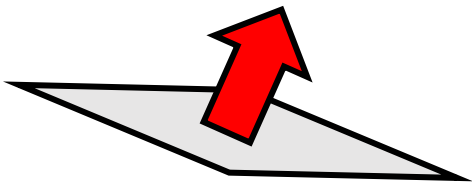
ex) 버튼을 눌렀는데 1이 증가했다.





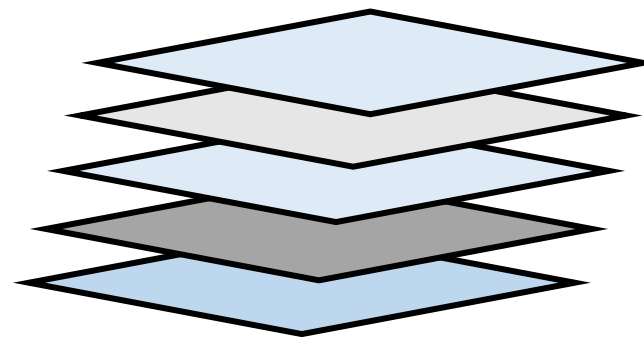
## Stack 구조

순서대로 쌓이고,  
맨 위에 있는 페이지가 눈에 보인다.



### Pop

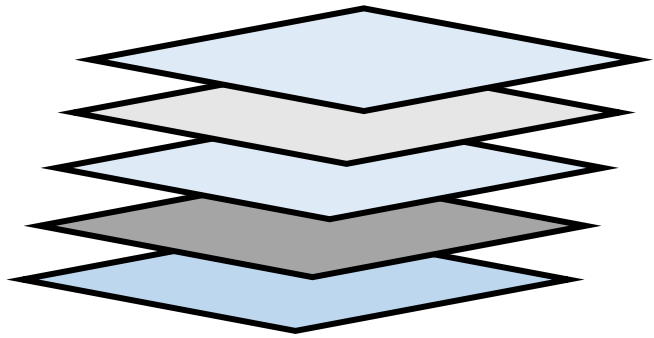
맨 위에 있는 페이지가  
삭제된다.



### Push

맨 위에 하나의 페이지가  
더 추가된다.

# ◀ 페이지



Push

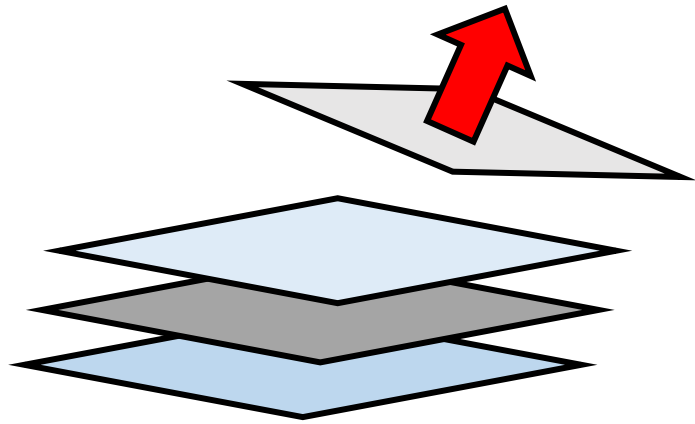
맨 위에 하나의 페이지가  
더 추가된다.



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

MainPage라고 정의되어있는 Class를  
맨 위에 덮어주는(Push) 함수

# 페이지



Pop

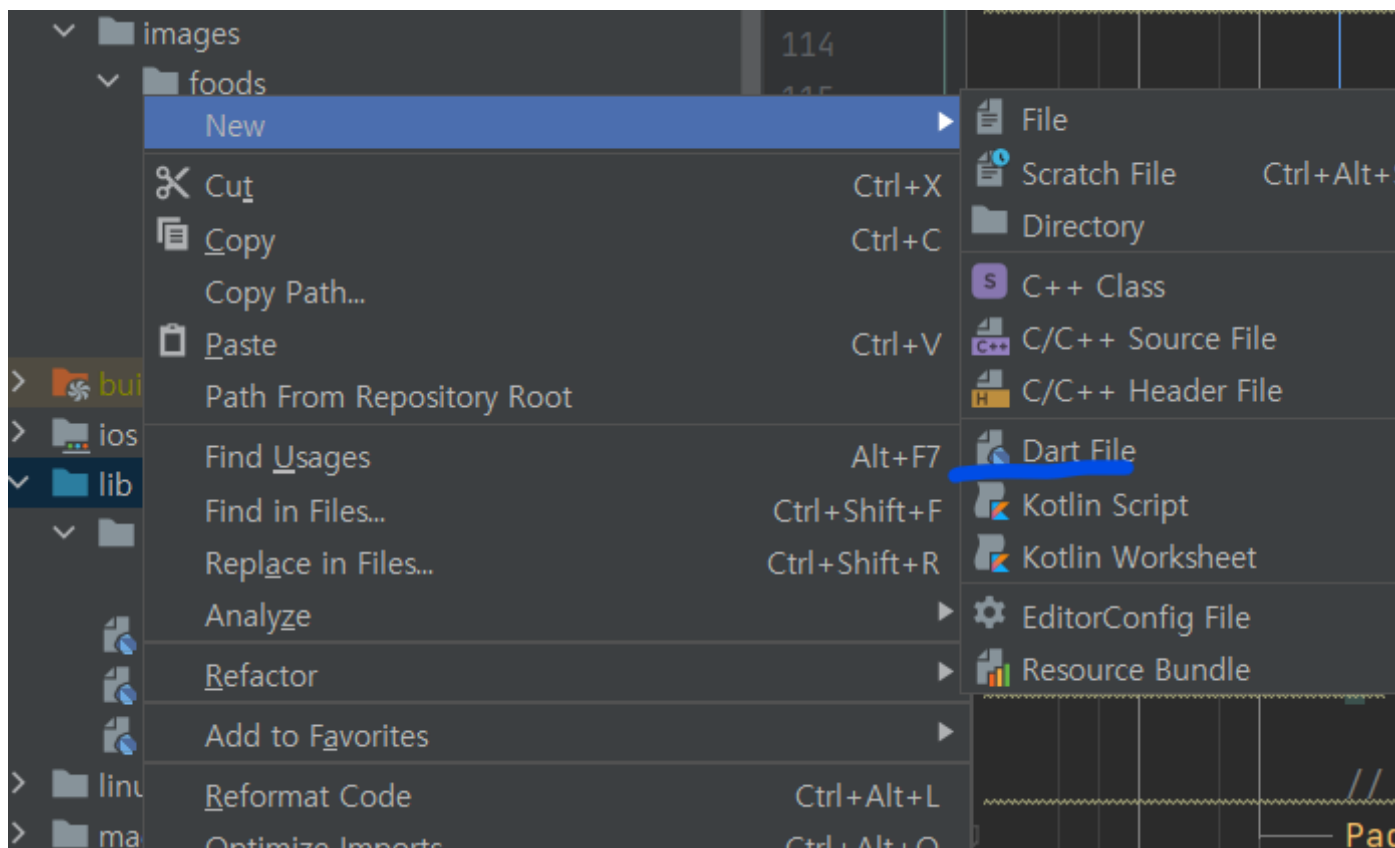
맨 위에 있는 페이지가  
삭제된다.

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

맨 위에 있는 페이지 한 장을  
없애주는 (Pop) 함수



# 새 페이지 만들기



Lib 폴더 하위에  
새 Dart 파일을 만든다.



# 새 페이지 만들기

```
1  st
   stless
   stful
   stanim
   const
   abstract
```

새 파일에 st를 입력하면  
자동완성 목록이 뜨는데, 알맞게 활용한다.

stless : StateLess  
stful : StateFul

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

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

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

틀이 자동으로 만들어지면

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

를 import해주어 코드에 뜨는 다양한 에러를  
해결해준다.



# Widgets

3일차 Flutter

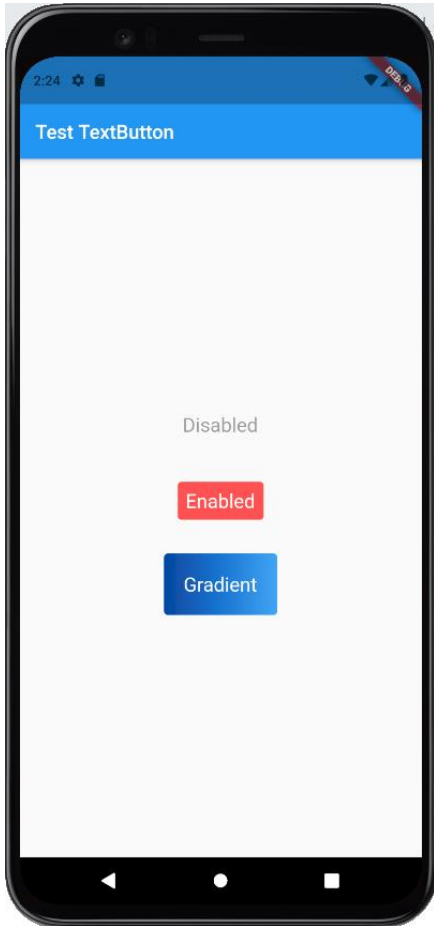


DGROID  
LEE HAIM



# 버튼과 함수

버튼에 기능을 넣자 !



```
TextButton(  
  onPressed: () {  
  
  },  
  child: Text("TextButton")  
),
```

```
onPressed: () {  
  print(' -- 룰렛 시작 !!');  
  
  setState(() {  
    counter ++;  
  });  
},
```

## onPressed

눌렀을 때 작동  
여기에 dart 코드를 넣을 수 있다.

Class간에는 쉼표(,)가 중요하지만,  
함수 내에는 코드 끝의 (;)가 필요하다.

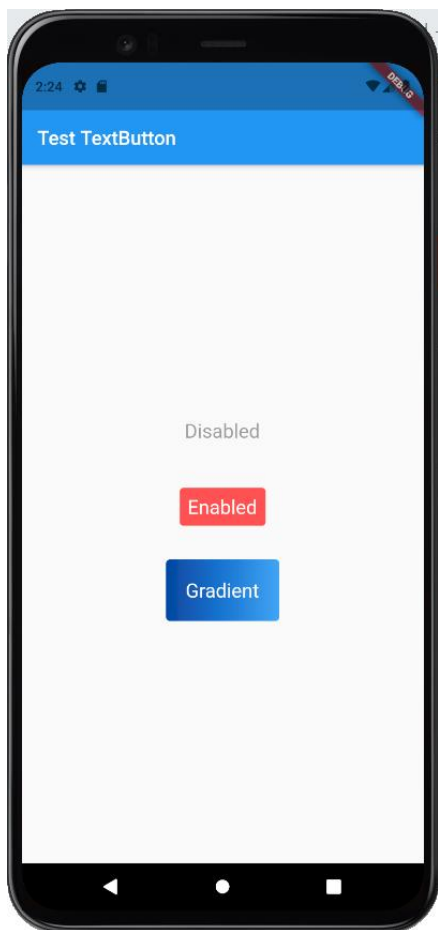
## onPressed 예시

- print문을 사용할 수 있다.
- setState() 함수를 사용하여 stateful인 경우, 화면을 바꿔줄 수 있다.



# 새 페이지로 이동하기

버튼을 사용하여 새 페이지로 이동하자 !



## TextButton

버튼 종류가 많으나 이거면 거의 다 해결됨

```
TextButton(  
  onPressed: () {  
  
  },  
  child: Text("TextButton")  
),
```

**onPressed**  
눌렀을 때 작동

**child**는 꾸미는 것  
옆의 코드는 TextButton이라는 텍스트가  
버튼을 꾸며주고 있다.

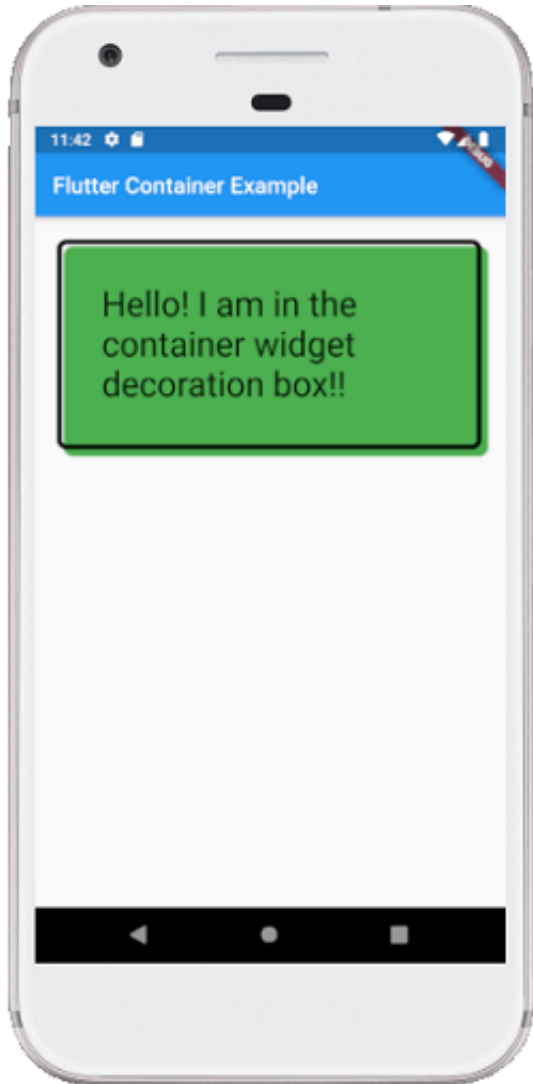
```
Navigator.push(context, MaterialPageRoute(builder: (_) => 만든페이지()));
```

onPressed 안에 넣어보자. (주의사항 : 자동완성을 사용하여 import를 할 것!)





# Container



## Container

children 안에 버튼, Container, 이미지 등 (**Widget**)을 넣을 수 있다.  
**Container**은 Widget들 중에서도 제일 간편하게 쓸 수 있음.

```
Container(),  
Container(child: ...)
```

child안에는 또 Widget이 들어갈 수 있다.



# Column/Row/Container

```
child: Column(  
  children: [  
    
```

## children

children 안에 버튼, Container, 이미지 등 (**Widget**)을 넣을 수 있다.  
**Container**은 Widget들 중에서도 제일 간편하게 쓸 수 있음.

```
// 맨 왼쪽  
Column(  
  children: [  
    FoodSVGPic(picName: "food1.svg",  
      size: 's', degree: 40,) // FoodS  
    FoodSVGPic(picName: "food2.svg",  
      size: 's', degree: -20,) // Food  
    FoodSVGPic(picName: "food3.svg",  
      size: 's', degree: 30,) // FoodS  
    FoodSVGPic(picName: "food4.svg",  
      degree: 15,) // FoodSVGPic  
    FoodSVGPic(picName: "food5.svg"),  
  ],  
) // Column
```

## Widget

괄호 짝 잘 맞추기

Widget 뒤에 **심표**가 안 붙어서 오류가 날 수 있다.

```
def add_a(x = 10, y = 20, z = 30) :  
    return x + y + z
```

**심표를 붙이는 이유**  
파이썬의 인수 전달 느낌

# Flutter 고수되기

Ctrl + 클릭

```
Container(),  
Container(child: )
```

Container을 이루는 내부 코드를 볼 수 있다.

```
class Container extends StatelessWidget {  
  /// Creates a widget that combines common painting, positioning,  
  ///  
  /// The `height` and `width` values include the padding.  
  ///  
  /// The `color` and `decoration` arguments cannot both be supplied  
  /// it would potentially result in the decoration drawing over the  
  /// color. To supply a decoration with a color, use `decoration:  
  /// BoxDecoration(color: color)`.  
  Container({  
    super.key,  
    this.alignment,  
    this.padding,  
    this.color,  
    this.decoration,  
    this.foregroundDecoration,  
    double? width,  
    double? height,  
    BoxConstraints? constraints,  
    this.margin,  
    this.transform,  
    this.transformAlignment,  
    this.child,  
    this.clipBehavior = Clip.none,  
  }) : assert(margin == null || margin.isNonNegative),  
       assert(padding == null || padding.isNonNegative),  
       assert(decoration == null || decoration.debugAssertIsValid(),  
       assert(constraints == null || constraints.debugAssertIsValid(),
```

다 해석하기는 어렵지만,  
변수 뜻을 해석하여 구글링을 하거나  
유추하여 값을 넣어볼 수 있다.



## 마우스 가져가기

Container에 관한 설명을 들을 수 있다.

- Container(),
- Container(child: ,)

```

package:flutter/src/widgets/container.dart

(new) Container Container({
  Key? key,
  AlignmentGeometry? alignment,
  EdgeInsetsGeometry? padding,
  Color? color,
  Decoration? decoration,
  Decoration? foregroundDecoration,
  double? width,
  double? height,
  BoxConstraints? constraints,
  EdgeInsetsGeometry? margin,
  Matrix4? transform,
  AlignmentGeometry? transformAlignment,
  Widget? child,
  Clip clipBehavior = Clip.none,
})

Containing class: Container

Creates a widget that combines common painting, positioning, and sizing widgets.
The height and width values include the padding.

The color and decoration arguments cannot both be supplied, since it would potentially result in
the decoration drawing over the background color. To supply a decoration with a color, use
`decoration: BoxDecoration(color: color)`.

```

Container의 변수, 필수 변수 등을 확인할 수 있다.

구글링과 엮어 모든 Class를 정복할 수 있다 !!