Flutter - Stateful widget

상태 플러터

위젯에 데이터를 저장하고 그 상태 변화를 보고 싶다면 stateful widget을 사용한다.

stateless는 그저 ui를 리턴할 뿐 데이터를 저장하고 있지 않다. 변경되지 않는 데이터를 보여주고만 있을 뿐이다.

Stateful widget

두 부분으로 나누어진다.

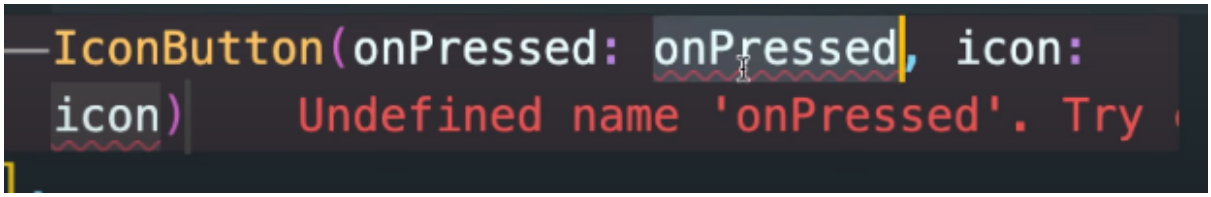
1. 상태가 없는 위젯
2. 위젯의 상태. state. 위젯에 들어갈 데이터와 UI를 넣는 곳

데이터가 변경되면 해당 위젯의 UI도 변경된다.

만드는 방법… 그냥 코드창에 st쳐서 새로 생성하거나, 클래스 만들때 extends 한 StatelessWidget 위에 마우스 올려서 컨트롤 . 하면 밑에 stateful로 바꾸는 액션이 뜸 그거 누르면 됨.

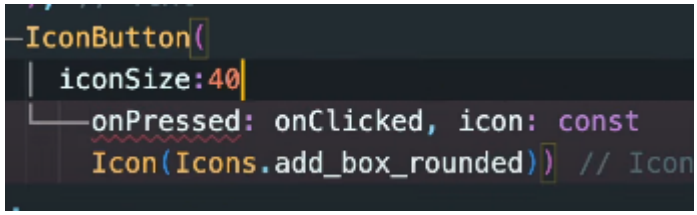
state가 변경될 때 UI는 새로고침되면서 최신 데이터를 보여준다.

flutter가 제공하는 아이콘 버튼



onPressed를 클릭할 때마다 실행할 함수를 만들어 할당하고

icon도 할당해야함~




```
void onClicked() {
  counter = counter + 1;
}
```

이렇게 하면 클릭할 때마다 counter가 1씩 증가하겠지요~~

하지만 되지 않음. 왜?

setState 함수가 없기 때문이지…

setState : State 클래스에게 데이터가 변경되었다고 알리는 함수

build 메서드를 새로 실행시킨다.

```
void onClicked() {
  setState(() {
    counter = counter + 1;
  });
}
```

이렇게 해줘야한다는 말이에요~

함수 안에 넣어야 가독성이 좋긴한데 아래처럼 적어도 실행 됨

```
void onClicked() {
  counter = counter + 1;
  setState(() {});
}
```

그치만 가독성이 죽어버렸으니 setState 함수 안에 넣도록 합시다.

그래서 반응형 UI를 만드는 법

1. Stateful widget을 만든다.
2. State에 위젯의 UI 관련 코드를 넣는다.
3. 위젯에 데이터를 넣는다.
4. 데이터가 변경되면 ui 새로 실행하라고 알려준다.

근데 플러터에서는 리액트보다 state를 많이 사용하진 않는대. 다른 방법이 많다고 해

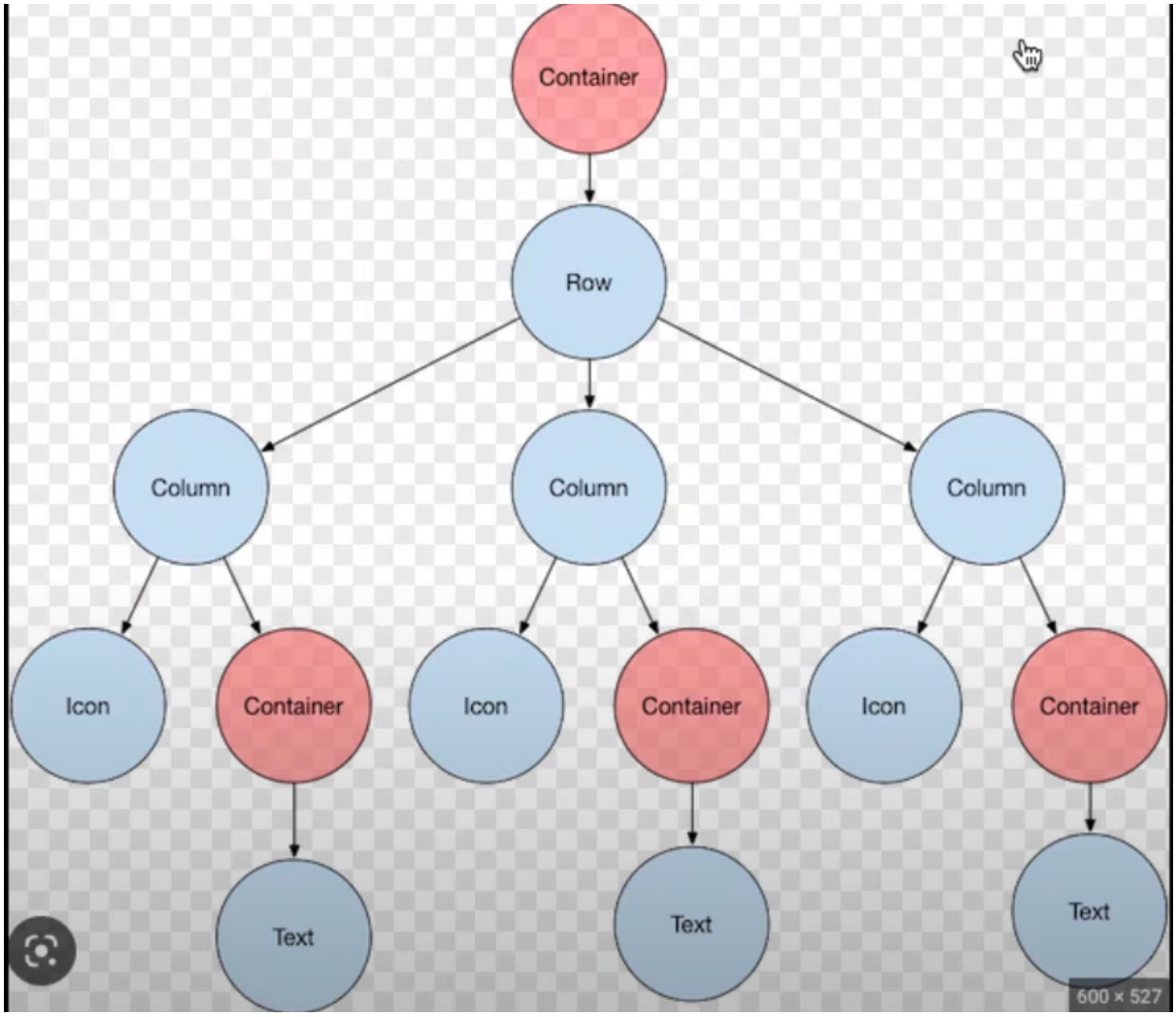
BuildContext

이전에 크기 색깔 하나하나들 요소마다 다 적용했는데, 플러터에선 그럴 필요가 없다. home:Scaffold 바로 위에 아래 코드를 적어주면 됨 *어플리케이션 위젯의 state에 있다는 점!!

```
theme: ThemeData(
  textTheme: const TextTheme(
    titleLarge: TextStyle(
  colorL Colors.red,
),),),
)
```

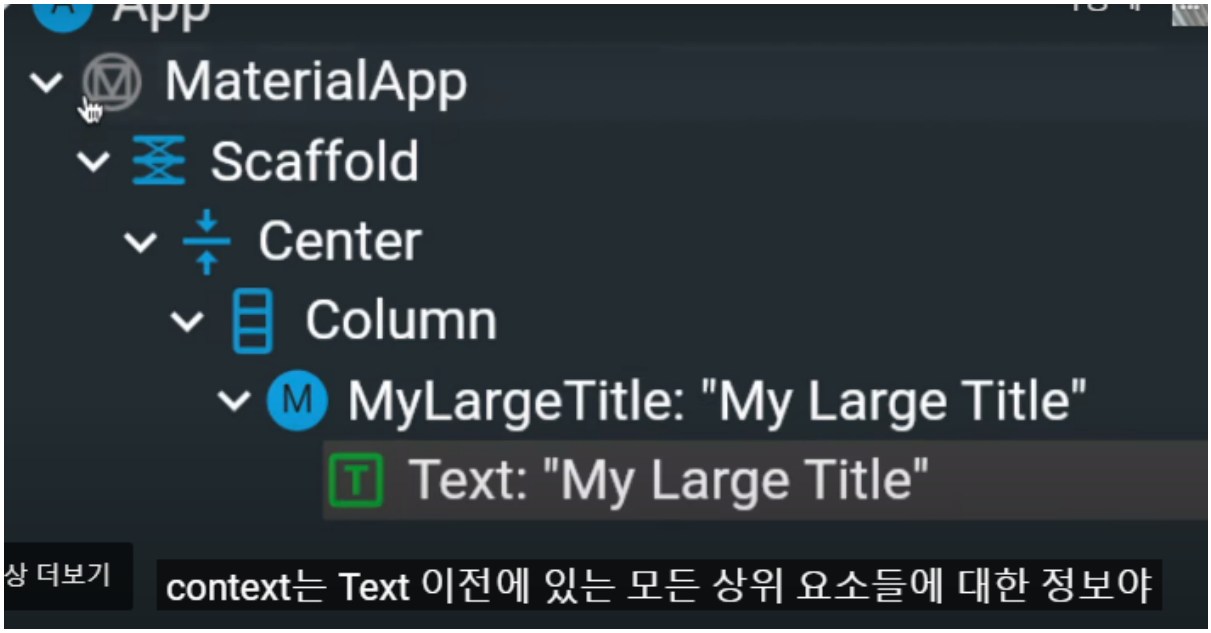
뭐 이렇게…

이제 이걸 사용하려면 theme에 접근해야함.



위 사진은 플러터가 애플리케이션을 어떻게 렌더링하는지 보여준다.

context : 이전에 있는 모든 상위요소들에 대한 정보.



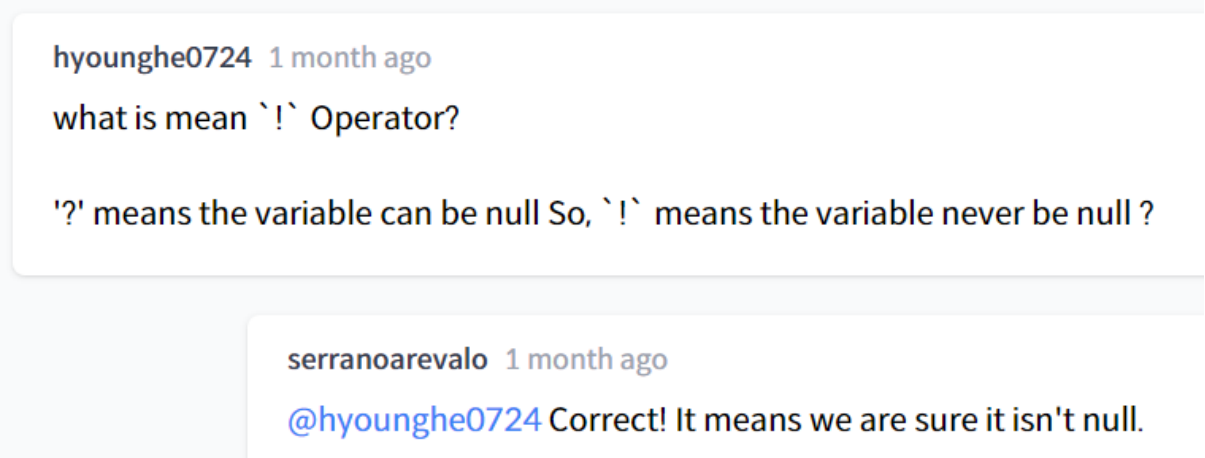
먼 조상의 데이터를 가져올 수 있다는 점에서 유용하다.

그럼 이제 부모요소에 접근해서 Theme를 불러올까

Theme.of(context).textTheme … 이렇게 파고파고 접근해

근데 이렇게만 하면 null safety가 작동해서 사용할 수 없다! 그래서 null이 아니라는 것을 알려줘야함!

그건…




```
Theme.of(context).textTheme.titleLarge!.color,)
```

! 로 확실하게 이야기해주던가 (아차 위로 올라가서 const 제거해야함)

```
Theme.of(context).textTheme.titleLarge?.color,)
```

? 로 있으면 사용하라고 해줄 수도 있음.

여기는 이해를 좀 해야겠는데…

of 는 현재 주어진 context에서 위로 올라가며 가장 가까운 [ ] 를 찾아 반환하라!

인 것 같습니다~

Widget Lifecycle

initState

stateful widget은 build 이외에 initState 메서드도 가지고 있다!

이것의 역할은 바로 상태를 초기화하기 위한 메서드

대부분의 경우 이것을 사용할 필요는 없다.

데이터 초기화 위해 context를 사용하거나 api에서 업데이트를 감지하거나

initState가 build보다 먼저 호출되고 initState는 오직 한 번만 호출된다.

```
@override
void initState(){
  super.initState();
  print("hello");
}
```

상태 변수는 밖에, 가끔 init을 사용해서 데이터를 초기화한다.

dispose

위젯이 스크린에서 제거될 때 호출되는 메서드다.

```
@override
void dispose(){
  super.dispose();
  print("dispose");
}
```

api 업데이트나 이벤트 리스너로부터 구독을 취소하거나 form에서 리스너를 취소하거나… 위젯이 위젯트리에서 제거하기 전에 뭘 취소하고 싶을 때 사용한다.

무슨 위젯을 토글해서 감추거나… 그러면 위젯 트리에서 제거되니까.