



1주차 복습

Dart?

→ flutter 에서 사용하는 언어!

- 자바와 비슷하게 VM 일 이용한 컴파일 방식.
- 따라서 어떤 플랫폼에서도 실행 가능!
- (개인적인 느낌으로) 자바와 스위프트를 적절하게 배합한 느낌!(에다가 코틀린 한방울)
- 즉, 태어난지 얼마안된 모던 언어

 Dart	<pre>class Segment { int links = 4; toString() => "I have \$links links"; }</pre>
 Kotlin	<pre>class Segment { var links: Int = 4 override fun toString()= "I have \$links links" }</pre>
 Swift	<pre>class Segment: CustomStringConvertible { var links: Int = 4 public var description: String { return "I have \(links) links" } }</pre>
 TypeScript	<pre>class Segment { links: number = 4 public toString = () : string => { return `I have \${this.links} links` }; }</pre>

수업내용

scaffold?

```
class BaseScaffold extends StatelessWidget {
  final String? title;
  final Widget? body;

  const BaseScaffold({
    Key? key,
    this.body,
    this.title,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(title ?? "unknown"),
      ),
      body: body,
    );
  }
}
```

가장 처음이자 아마 많이 볼 수 있는 위젯, scaffold라는 단어는 우리에게 많이 생소하지만, 발판이란 뜻을 가지고, 실제로 가장 베이스가 되는 위젯.

카드보드 같은 위젯사용.

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

  @override
  Widget build(BuildContext context) {
    return BaseScaffold(
      title: "김재하",
      body: SizedBox(
        width: double.infinity,
```

```

child: Column(
  crossAxisAlignment: CrossAxisAlignment.center,
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    const Text(
      "chatting",
      style: TextStyle(
        color: Colors.deepOrange,
        backgroundColor: Colors.greenAccent,
        fontSize: 50,
        fontWeight: FontWeight.bold,
      ),
    ),
    const Padding(
      padding: EdgeInsets.only(top: 30, bottom: 50),
    ),
    ...
  ],
)

```

기본적으로, 화면에 띄어지는 모든 화면은 위젯이며, flutter는 이런 위젯을 쌓아올리는 방식으로 화면을 구성하는 방식.

final과 const

```

final String? title;
final Widget? body;
...
children: [
  const Text(
    "chatting",
    style: TextStyle(
      color: Colors.deepOrange,
      backgroundColor: Colors.greenAccent,
      fontSize: 50,
      fontWeight: FontWeight.bold,
    ),
  ),
  ...
]

```

컴파일러의 바뀌지 않는 부분에 대한 쓸데없는 연산을 줄이기 위해, 화면상에 더이상 바뀌지 않는 부분에 대한 요소를 바뀌지 않는 상수(const, final)로 만들어 줌으로서, 최초 1회 컴파일 이후, 이부분의 코드가 변경되지 않는이상 컴파일을 생략해서 성능을 향상시킨다.

? 과 ?? 연산자

```

final String? title;
final Widget? body;
...

```

```
title: Text(title ?? "unknown"),  
...
```

null이란 값이 존재하지 않는 dart언어의 특성상, 하나의 변수에는 무조건 값이 들어가야한다. 하지만 그 값을 나중에, 혹은 지정하지 않아야할 경우를 위해, 사용되기전까지, 이 변수에 값이 들어있는지 없는지 모르는 상태인 '?'를 추가한다. 이 ?의 특징을 이용하여 변수안의 값 존재 유무에 따라 값을 출력할지, 아님 또다른 statement 를 실행할지 결정하는 ??연산자가 존재한다.

다른 형태의 위젯들

- Column, row : 행, 열 자신들의 child widget들을 행, 열 처럼 놓는 위젯.

- 버튼 :

1. ElevatedButton -> 버튼모양이 있는 버튼(애니메이션 있으므로 const 불가)
2. TextButton → 텍스트모양 버튼

- SingleChildScrollView

화면을 넘는 위젯이 있는 경우, 스크롤을 지원하여 아래로 내릴 수 있게 만드는 위젯

stateless, stateful

1. stateless : 어떤 위젯에 대한 state가 변화하지 않는 위젯
2. stateful : 어떤 위젯에 대한 state의 변화가 관찰되어, 그에 대한 화면 변화등과 값은 변화가 발생하는 위젯

switch 를 이용하여 화면 전환하는 세가지 방법

```
Switch(  
  value: switchValue,  
  onChanged: (value) {  
    switchValue = !switchValue;  
  }  
)
```

```

        print(switchValue);
        setState(() {});
    },
),
switchValue ? build4column() : build8column(),
if (!switchValue) build4column(),
if (switchValue) build8column(),
(() {
    Widget? widget;
    if (switchValue) {
        widget = build4column();
    }
    if (!switchValue) {
        widget = build8column();
    }
    return widget!;
})()

```

1. 삼항 연산자 이용
2. if문 활용
3. 함수 활용(익명함수)

세번째 함수 작성시 유의사항.

리턴할 함수는 값이 무조건 존재해야함! → ?함수를 그대로 쓰면 안됨!

따라서,

widget 변수를

1. 리턴시 !를 붙여 분명한(?) 변수로 만들어줘야함
2. 애초에 값을 가지도록 설정
3. lazy

중 하나로 설정해 줘야함!