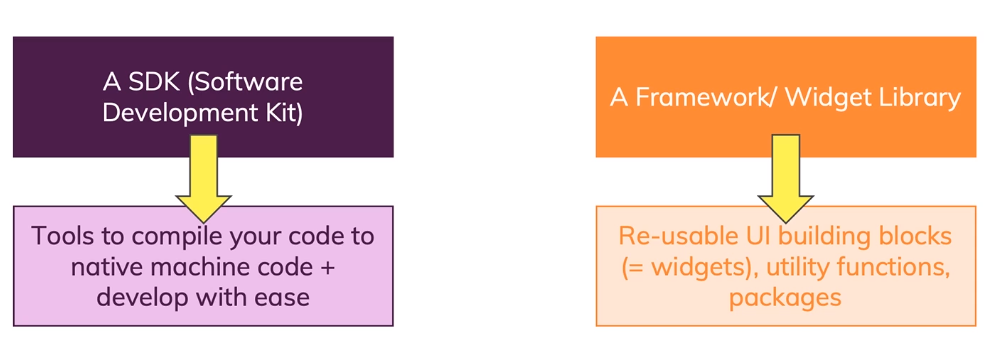
10-03-21

Flutter

Framework

It is a tool that allows you to build native cross-platform apps with one programming language and codebase.

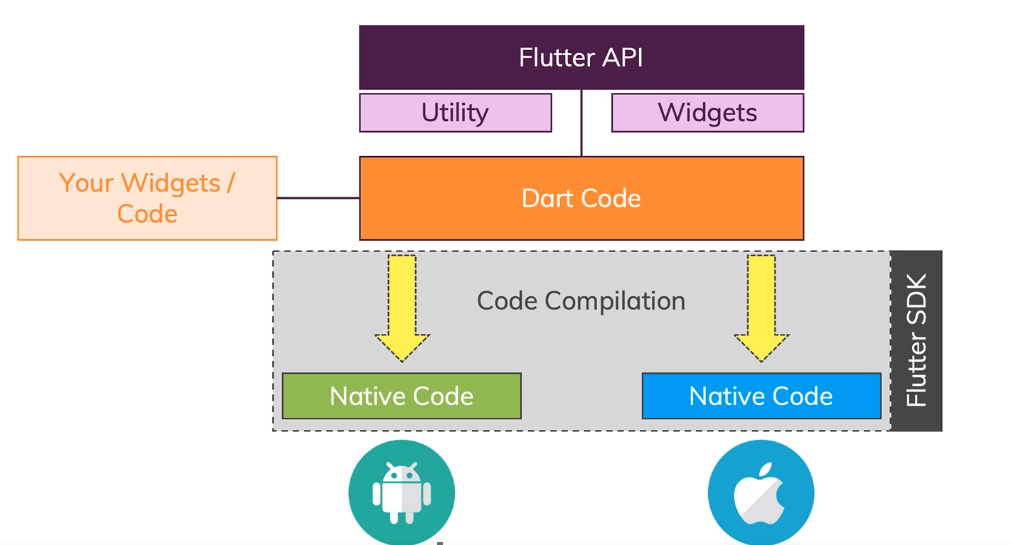


Flutter Architecture



All about widgets

Working of Flutter



Flutter compiles its own pixels, whereas other frameworks use platforms. This gives a flutter advantage in performance.

Material Design

Material Design: Buttons and others Ui in a google style

Cupertino: in Apple style

When we create a new application in a flutter (Ctrl + Shift + P), you get a code something like this

main() function only used to run the app

runApp(class name instance)

import 'package:flutter/material.dart';

void main(List<String> args) {

runApp(MyApp());

}

class MyApp extends StatelessWidget { *// ! StatelessWidget has its build method with parameter as(context)*

@override

Widget build(BuildContext context) { *// ! we are overriding the build method,*

return MaterialApp(

home: Scaffold(

body: Column(

),

));

}

}

Basic app

import 'package:flutter/material.dart';

void main() {

runApp(HomePage());

}

class HomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Text('Hello!'),

);

}

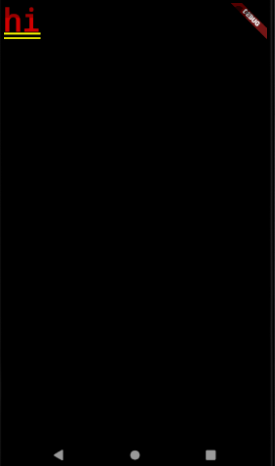
}

Materials App provides a built-in class

StatelessWidget has a build method.

Build method builds the app.

In the main function to run the app runApp(class new instance());



The Scaffold provides a basic UI

Inside Scaffold, we create new Widgets

class HomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("My First App"),

),

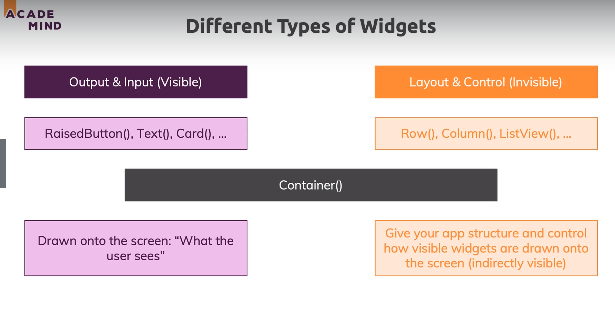
body: Text('hi hello'),

));

}

}



Body only takes one widget, so to have multiple widgets we use these

body: Column(

children: <Widget>[

Text('The questions!'),

RaisedButton(

child: Text('Answer 1'),

onPressed: answerQuestions,

),

RaisedButton(

child: Text('Answer 2'),

onPressed: () {

print('Answer 2 Chosen');

},

),

RaisedButton(

child: Text('Answer 3'),

onPressed: () => print('Answer 3 Chosen'),

),

],

)

Column is layout

children: <Widgets>[]; //takes lists as input

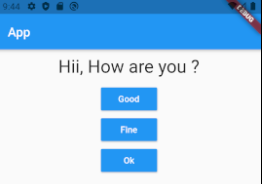
RaisedButton // takes two arguments 1)child : Text(‘’),

2)onPressed : //mention a func

//Anonymous function

onPressed :() {

print();  
}



To change the text we need array type data structures so we use list

var \_questionsIdx = 0; *//\* store index or itarator*

var \_questionsList = [

'Hii, How are you ?',

'What’s Your age ?',

];

Replace text widget

Text(\_questionsList[\_questionsIdx]),

Write a function to change question if click on any button

void answerQuestions() {

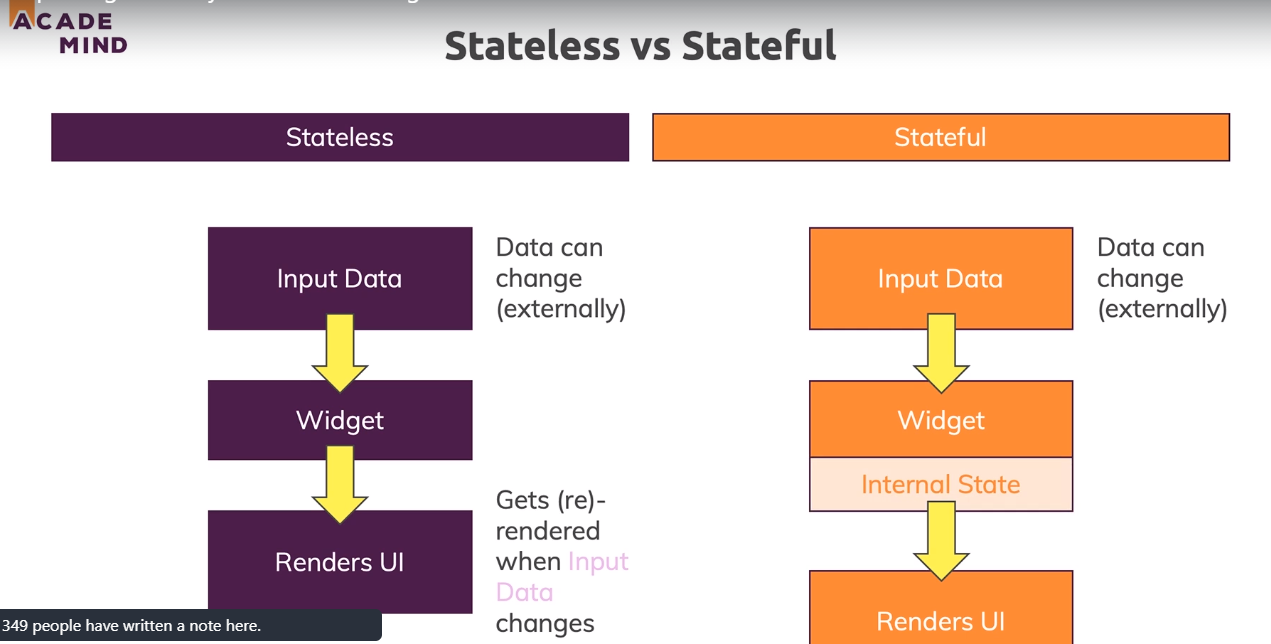
if (questionsidx < 1) {

questionsidx = questionsidx + 1;

}

});

If click on button, question wont change because in stateless function



class HomePage extends StatefulWidget {

@override

State<StatefulWidget> createState() {

*// TODO: implement createState*

return HomePageState();

}

}

class HomePageState extends State<HomePage> {

int questionsidx = 0; *//\* store index or iterator*

void answerQuestions() {

setState(() {

if (questionsidx < 1) {

questionsidx = questionsidx + 1;

}

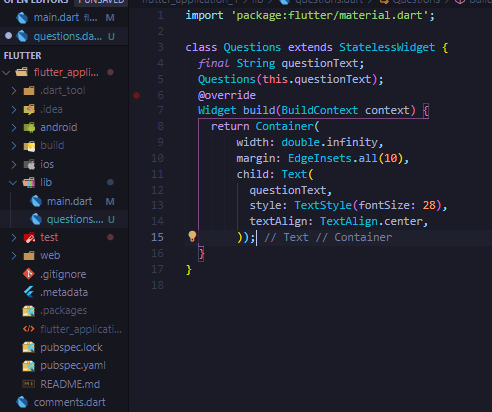
});

print(questionsidx); }

setState() calls build methods , and checks in which function is varying in build func and builds its with new info,to increase performance

11-03-21

Always use 1 widget one build file, it would keep the main file small and clean and even easy to make changes //Questions



To make Text,bigger we make TextStyle(fontSize :28),

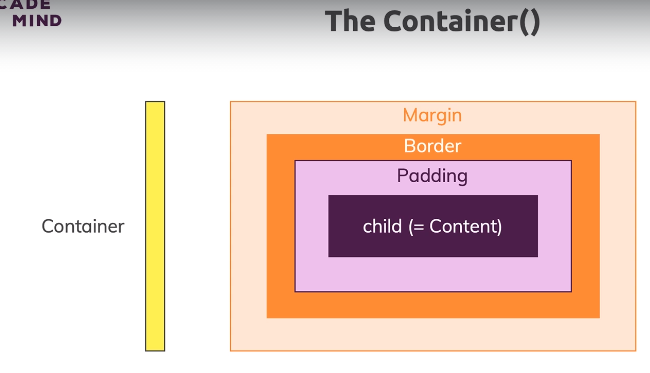
Now to make

textAlign : TextAlign.center,

but to show changes we have to make a container with

width: double.infinty,

Container is a widget which use mainly used for styling , it has many function which width, height , margin, padding ,decoration and many more



Creating a Answers buttons separate file, as we dont want hardcode many buttons so we just call this function its creates buttons according to the list

class Answers extends StatelessWidget {

*final* Function questionchanger;

*final* String answers;

Answers(this.questionchanger, this.answers);

@override

Widget build(BuildContext context) {

return Container(

width: double.infinity,

child: RaisedButton(

child: Text(answers),

textColor: Colors.white,

onPressed: questionchanger,

color: Colors.grey.shade800,

),

);

}

}

Creating a list with maps so we can change the button’s text dynamically,

var questions = [

{

'questionText': 'What\'s your favorite color ?',

'answers': ['Black', 'Red', 'Green', 'White'],

},

{

'questionText': 'What\'s your favorite animal ?',

'answers': ['Rabbit', 'Dog', 'Cat', 'Lion'],

},

{

'questionText': 'What\'s your age ?',

'answers': ['<18', '18-21', '21-30', '>30'],

},

]; *//\* list of maps to store questions*

To link these to answers buttons , we traverse using a new function

children: <Widget>[

Questions(questions[\_questionsidx]['questionText']),

...(questions[\_questionsidx]['answers'] as List<String>) //as list as cant find its list

.map((answers) {

return Answers(answerQuestions, answers);

}).toList()

],

We use this func as List<String> to tell flutter its a list

[map](https://www.youtube.com/watch?v=lisp9pBjcGs&ab_channel=OvidiusMazuru) function for lists is like a forEach func but it return a Iterable to convert in into list we can use .toList() func ,

var New = list.map((e){// here New is MappedListIterable<dynamic, dynamic>

print(e);

return e;

});

List New = list.map((e) { here New is list

print(e);

return e;

}).toList();

But in our app code, it would return a list of widgets , but in column we need in widgets , so we use “...” function to break the list into objects,

**final** is runtime constant

**const** is compiled time constant

1.const list =[] vs 2. var list = const[ ]

1. list and data is protected, whereas 2) only data is protected

We can assign list=[], it is not possible in case 1.

Result Screen when all three questions are done, so we in again create list of maps inside ‘answers’ with scores

class \_HomePageState extends State<HomePage> {

int \_questionsidx = 0;

var \_totalScore = 0;

*final* \_questions = *const* [

{

'questionText': 'What\'s your favorite color ?',

'answers': [

{'text': 'Black', 'score': 10},

{'text': 'Red', 'score': 15},

{'text': 'Green', 'score': 17},

{'text': 'White', 'score': 20},

],

},

{

'questionText': 'What\'s your favorite animal ?',

'answers': [

{'text': 'Rabbit', 'score': 10},

{'text': 'Dog', 'score': 10},

{'text': 'Cat', 'score': 10},

{'text': 'Lion', 'score': 10},

],

},

{

'questionText': 'What\'s your age ?',

'answers': [

{'text': '<18', 'score': 10},

{'text': '18-21', 'score': 10},

{'text': '21-30', 'score': 10},

{'text': '>30', 'score': 10},

],

},

];

Now we create a separate Result file which shows a text and score,

so

class Result extends StatelessWidget {

*final* int score;

*final* Function reset;

Result(this.score, this.reset);

String get resultPhrase {

var resultText = 'You did it! ';

return resultText + 'Your Score is $score';

}

@override

Widget build(BuildContext context) {

return Center(

child: Column(

children: <Widget>[

Text(

resultPhrase,

style: TextStyle(fontSize: 36, fontWeight: FontWeight.bold),

textAlign: TextAlign.center,

),

TextButton(

onPressed: reset,

child: Text("Resart Quiz"),

style: ButtonStyle(

foregroundColor:

MaterialStateProperty.all(Colors.deepPurple)))

],

),

);

}

}

Later in main file we right logic for reset button that , and create new variable for total score

var \_totalScore = 0;

When we click on reset button it changes the values of

void \_resetQuiz() {

setState(() {

\_questionsidx = 0;

});

\_totalScore = 0;

}

Now to call this result we use a ternary operator inside body of Scaffold

body: \_questionsidx < \_questions.length

? Quiz(\_answerQuestions,\_questions, \_questionsidx)

: Result(\_totalScore, \_resetQuiz)));

void \_answerQuestions(int score) {

\_totalScore += score;

setState(() {

if (\_questionsidx < \_questions.length) {

\_questionsidx = \_questionsidx + 1;

}

});

print(\_questionsidx);

}