Sử dụng list trong dart:

List<data\_type> name = [];

or var name= [];

Thêm dữ liệu: nameList.add(gtri);

Thêm toàn bộ gtri của list khác: nameList.addAll(tên list khác);

Thêm gtri vào vị trí x: nameList.insert(vị trí x, gtri);

Xóa gtri muốn xóa: nameList.remove(gtri muon xóa);

Xóa tại vị trí: nameList.removeAt(vị trí muốn xóa);

Xóa từ vị trí x đến y: nameList.removeRange(x,y);

Xóa tất cả: nameList.clear();

Đảo lại vị trí gtri trong danh sách: nameList.reversed.forEach((i))…

Sử dụng maps trong dart:

Khai báo : var nameMaps = {}; gtri trong map gồm 2 phần: tên : gtri

Có thể thêm luôn gtri vd: var nameMaps = {‘id’:1 , ‘ten’:’hung’};

Cách in maps: nameMaps.forEach((name,value){print(‘$name-$value’)});

Thêm phần tử: nameMaps[‘tên’]=’gtri’ ;

C ý: nếu tên trùng nhau khi thêm gtri sẽ ghi đè

Lấy ra 1 phần tử: print(nameMaps[‘tên’])

Xóa gtri: nameMaps.remove(‘tên’);

Ktra xem có phần tử nào tên tồn tại không: bool check = nameMaps.containsKey(‘ten’);

Ktra xem có phần tử nào gtri tồn tại không: bool check = nameMaps.containsKey(‘gtri’);

Sử dụng Sets trong dart: giống maps nhưng sets chỉ có phần gtri không có tên

Sử dụng Queue trong dart:

Khai báo: Queue<data\_type> nameQueue = Queue()

Biểu thức điều kiện:

Condition ? expr1 : expr2

Name = check ?? ‘default’ ;

Nếu check là null thì gán name là default

Có thể viết : number..adđ(1)..add(2) thay cho number.add(1);number.add(2);

|  |  |  |
| --- | --- | --- |
|  | StatelessWidget | StatefulWidget |
| override | build | createState->build |
|  |  |  |
|  | Không thể thay đổi biến truyền vào | Có thể thay đổi biến truyền vào |
|  |  |  |

import 'package:flutter/material.dart';  
  
void main() {  
 runApp( MaterialApp(  
 home : SafeArea(  
 child: Scaffold(  
 body: Center(child: MyWidget2(false)),  
 // appBar: AppBar(backgroundColor: Colors.red,  
 // title: Text('tu hoc flutter'),  
 // ),  
 //  
 // body:Center(  
 // child: Text('hello world'),  
 //  
 // ),  
  
  
  
 ))  
  
 ));  
   
}  
class MyWidget extends StatelessWidget{  
 final bool loading;  
  
  
 MyWidget(this.loading);  
  
 @override  
 Widget build (BuildContext context) {  
 if (loading){  
 return const CircularProgressIndicator();  
 }  
 else return const Text('state');  
 }  
   
}  
class MyWidget2 extends StatefulWidget{  
 final bool loading;  
  
 MyWidget2(this.loading);  
  
 @override  
 State<StatefulWidget> createState() {  
 return MyWidget2state();  
 }  
  
}  
class MyWidget2state extends State< MyWidget2>{  
 late bool \_localLoading;  
 @override  
 void initState() {  
 \_localLoading=widget.loading;  
 }  
 //hàm init dùng để khởi tạo gtri ban đầu và chạy trước hàm build  
 @override  
 Widget build(BuildContext context) {  
 if (\_localLoading){  
 return const CircularProgressIndicator();  
 }  
 else return FloatingActionButton(onPressed: onClickButton);  
 }  
 void onClickButton(){  
 setState(() {  
 \_localLoading=true;  
 });  
 }  
 }

class MyWidget extends StatelessWidget{  
  
 @override  
 Widget build (BuildContext context) {  
  
 return const Text('Nếu bạn tạo bất cứ biến nào là final, bạn không thể thay đổi'  
 ' giá trị của biến final (nó sẽ là hằng số). Ví dụ của biến final '  
 'trong Java: Giả sử có một biến final là MAX\_SPEED. Bạn cố ý thay đổi '  
 'giá trị của biến này nhưng nó không bị thay đổi, bởi vì biến final '  
 'một khi được gán giá trị thì không bao giờ thay đổi được.',  
 textDirection: TextDirection.ltr, //hướng text từ trái sang phải  
 textAlign: TextAlign.left, //căn lề  
 maxLines: 3, //giới hạn số dòng  
 overflow: TextOverflow.ellipsis, //hiển thị ... ở cuối  
 textScaleFactor: 1,  
 style:TextStyle(  
 // color: Colors.pink,  
 // backgroundColor: Colors.yellowAccent,  
 // fontWeight: FontWeight.w400, //độ đậm nhạt của chữ  
 //fontFamily:'DFVN ED Lavonia',  
 wordSpacing: 20, //khoảng cách giữa các từ  
 letterSpacing: 1, //khoảng cách giữa các chữ  
 // decoration: TextDecoration.overline, //gạch chân  
 )  
 );  
   
}  
  
}  
class MyWidget1 extends StatelessWidget{  
 @override  
 Widget build(BuildContext context) {  
 return Center(  
 child: RichText( // dùng nhiều kiểu font trên 1 text  
 text: TextSpan(  
 style: DefaultTextStyle.*of*(context).style,  
 children: const <TextSpan>[  
 TextSpan(text: 'hello'),  
 TextSpan(text: 'bold',style: TextStyle(fontSize: 15,fontWeight: FontWeight.*bold*)),  
 TextSpan(text: 'world !!!')  
 ]  
 )  
 ),  
 );  
 }  
  
}  
class MyWidget2 extends StatelessWidget{  
 @override  
 Widget build(BuildContext context) {  
 return const Card(  
 margin: EdgeInsets.all(20.0), //cách lề xung quanh 20 đv  
 color: Colors.*pink*,  
 child: Padding(  
 padding: EdgeInsets.all(30.0), // text sẽ cách đều tất cả các phía 30 đv  
 // .from LTRB cho phép căn chỉnh từng phía  
 //.only(left:20.0) cho phép cách trái 20 đv  
 //symmetric(vertical:100.0,horizontal:100.0) cách dọc và ngang 100 đv  
 child: Text('hung',  
 style: TextStyle(fontSize: 20,  
 color: Colors.*white*),  
 ),  
 ),  
 );  
 }