Flutter课程02：Dart 语法基础

一、课前准备

- 观看预习视频并安装好Dart环境

二、课堂目标

- 用更适合web前端的方式带大家迅速上手Dart语言

三、知识要点

- Dart语言的基本数据类型
- 类型判断，类型转换，级联操作
- 集合类型
  - List
  - Set
  - Map
- 函数的定义
  - 可选参数
  - 命名参数
  - 箭头函数
  - 异步Async，Await，Future类型
- 类
  - 构造函数
  - 内部，外部定义，静态

    - extends，implement，with
- 外部库，内置库
- 库的重命名，部分导入

四、总结

- Dart语言规范已被纳入ECMA Script体系，熟练掌握JS的同学上手并不难，但差异比如会导致增加一些记忆成本。

五、作业

- 实现一个抽象类Vehicle，定义数值车轮wheels，定义抽象方法getWheelsNum。实现一个汽车类继承自Vehicle。新建汽车实例调用getWheelsNum打印日志wheels数值。

复习和解答上节课的遗留问题

1. setState的问题
2. 使用 AnimatedPositioned 改进一下上节课的代码

> Dart语言规范已被纳入ECMA Script体系，熟练掌握JS的同学上手并不难，但差异比如会导致增加一些记忆成本。

Dart的基本数据类型

- 数值类型 int,double
- 字符串 String
- 布尔 bool
- 数组 List
- 字典 Map

还有一些不常用的

Dart使用严格的分号

使用 var 自动推断类型；

使用类型严格定义；

使用 Print 打印日志

数值类型

```
int var1 = 123;
double var2 = 12.3;
```

运算符

~/ 取整

```
int a=15;
int b=2;
print(a%b);   //取余 1
print(a~/b);  //取整 7
print(a/b); //  7.5
```

类型转换

int.parse , double.parse

```
String var1 = "1.1";

print((double.parse(var1)));
```

定义String类型

```
var str = "字符串";

print(str)
```

三个引号 ''' 来定义多行字符串，三个双引号也是

```
String str = '''
字符串
字符串
字符串
'''
```

字符串拼接

```
String str1 = "我爱";
String str2 = "学习";

print(str1+str2);
```

字符串+数组

```
String str1 = 123;
String str2 = "学习";

print("$str1 $str2");
```

表达式

```
print("${表达式}");
```

布尔 bool

类型判断 is

```
value is int   //返回 true/false
```

类型获取

```
.runtimeType
```

数组 List

```
var mylist = [1,2,3,4,5];

List mylist = [1,2,3,4,5];

var mylist = new List();

mylist.add("aaa");
```

```
//指定类型集合
List<int> mylist = [];

var mylist = new List<int>();
```

常用方法

```
add
addAll                                  两个List合并
indexOf
remove
removeAt
removeLast()                            删除末尾元素
removeRange(start,end)                  范围删除
removeWhere()                           根据条件删除
fillRange(start,end,value)              从start-end
每个元素用value替换
insert(index,value)
insertAll(index,list)
setAll(index,list)                  从index开始，逐个替换
list中的元素
mylist.toList(growable:false)       生成一个新数组，
growable
first,last
join()
split()
map()                       遍历现有List的每个元素，并做处
理，返回一个新的Iterable
reversed                获取反向的Iterable
shuffle()
```

take(count) 从0开始取count个元素，并返回结果
takeWhile((e)=>(bool)) 从0开始取，直至第一个不符合函数的元素，将其前面的元素都返回。

retainWhere((element)=>(element>2)); 根据条件保留

```
//判断
contains(element)
where((element)=>(element>2));
any((element)=>(element>2));
every((element)=>(element>2));

//查找
forEach((element){})
```

firstWhere((element)=>(bool)) 返回第一个满足条件的元素（不是元素的index）
lastWhere((e)=>(bool)) 从后往前找，返回第一个满足条件的元素的值(不是index)
indexWhere((e)=>(bool)) 返回第一个满足条件的元素的index
lastIndexWhere((e)=>(bool)) 从后向前找 返回第一个满足条件的元素的index

toSet() 转为Set
asMap() 转为Map

Set （参考ES6的Set）

s.toList

字典 Map

```
var myMap = {};

Map myMap = {};

var myMap = new Map();

Map myMap = {
    "name":"哈哈哈"
};

print(myMap["name"]);
```

常用属性和方法

```
常用属性:
    keys            获取所有的key值
    values          获取所有的value值
```

```
    isEmpty          是否为空
    isNotEmpty       是否不为空
常用方法:
    remove(key)      删除指定key的数据
    addAll({...})    合并映射   给映射内增加属性
    containsValue    查看映射内的值   返回true/false
    forEach
    map
    where
    any
    every
```

数据循环的方法

```
// Map person={
  //   "name":"张三",
  //   "age":20
  // };


  // var m=new Map();
  // m["name"]="李四";

  // print(person);
  // print(m);

//常用属性:
```

```
    // Map person={
    //   "name":"张三",
    //   "age":20,
    //   "sex":"男"
    // };

    // print(person.keys.toList());
    // print(person.values.toList());
    // print(person.isEmpty);
    // print(person.isNotEmpty);


//常用方法:
    Map person={
      "name":"张三",
      "age":20,
      "sex":"男"
    };

    // person.addAll({
    //   "work":['敲代码','送外卖'],
    //   "height":160
    // });

    // print(person);
```

```
    // person.remove("sex");
    // print(person);



    print(person.containsValue('张三'));
```

定义方法

```
  void func(){}

  int getNum(){}
```

参数

```
//参数
func(name)
//强制类型
func(String name)
//默认值
func(String name='name')
//可选参数
func(name,[age,gender])
//命名参数
func(name,{age,gender})
```

箭头函数（和JS不一样,dart的箭头函数只能写一“行”）严谨的说法1个表达式

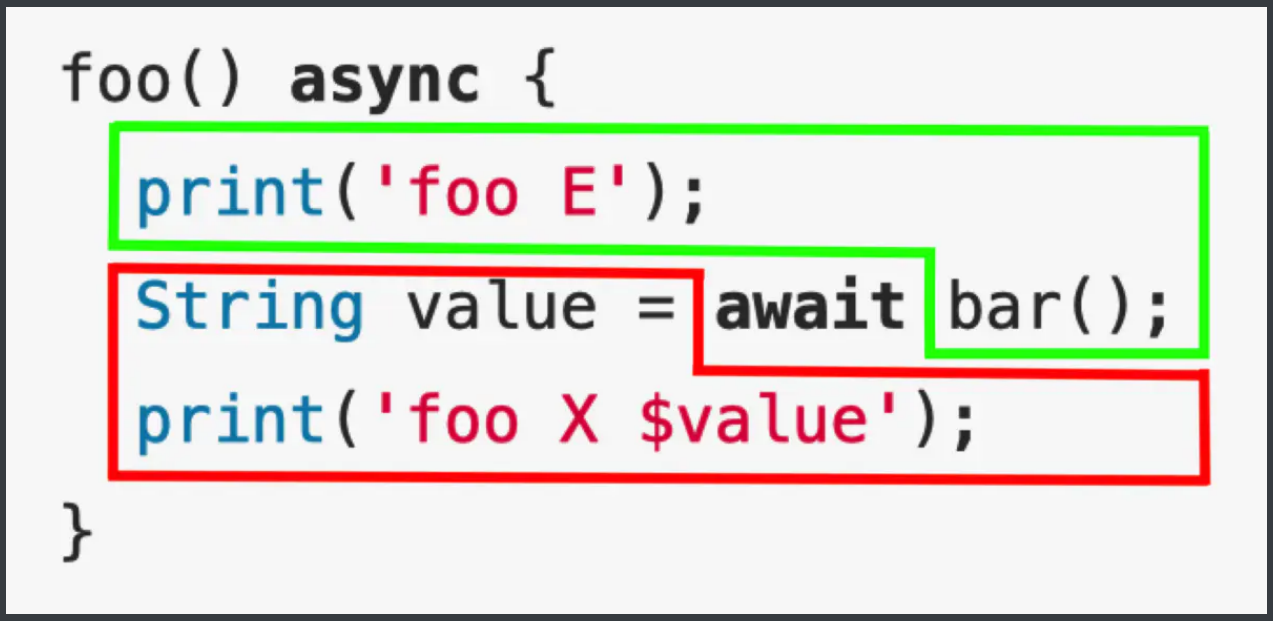
```
myList.map((value)=>print(value));

myList.map((value)=>{
  //只能一行
});
```

自执行方法 和 js一样

```
(){

 })();
```

异步Async，Await，Future类型

```
foo() async {
  print('foo E');
  String value = await bar();
  print('foo X $value');
}

bar() async {
  print("bar E");
  return "hello";
}

main() {
  print('main E');
  foo();
  print("main X");
}
```




```
foo() {
  print('foo E');
  return Future(bar).then((value) => print('foo X
$value'));
}
```

类class

```
class Person{
  String name ="";
  int age = 12;
  int gender = 1;
  //默认是public, 加_变为私有
```

```
  // String _name;

  //构造函数
  Person(String name,int age){
    this.name = name;
    this.age = age;
  }

  /*
  命名构造函数
  Person.init(){

  }
  */
  /*
  初始化构造
  Person():name='',age=12{

  }
  */

  void info(){
    print("$name,$age,$gender");
  }

  /*
  get info{
    return "$name,$age,$gender";
  }
```

```
  */

  void setName(String name){
    this.name = name;
  }
  set personName(String name){
    this.name = name;
  }

  //静态方法，静态属性
  //static
  //非静态方法可以访问静态或非静态成员
  //静态方法只可以访问静态成员
}
```

连缀操作 ..

```
Person p1 = new Person();
//不要加分号
p1..name=
  ..age=
```

继承 extends

```
class Animals
```

```
{
  void eat(){
    print("吃");
  }
}

class Dog extends Animals
{
  @override //可以不写
  void eat(){
    print("吃骨头");
  }
}
class Cat extends Animals
{
  @override
  void eat(){
    print("吃鱼");
  }
}
```

抽象类 abstract

抽象类通过abstract关键字来定义

抽象类中不实现方法体的方法就叫抽象方法

如果某个类extends一个抽象类，则必须实现抽象类中的抽象方法

如果某个类implements一个抽象类,则必须实现抽象类中的所有属性和方法

抽象类不能被new（实例化）,只有继承或实现它的子类可以

- 用法

如果要复用抽象类里的方法，并且希望用抽象方法约束子类实现，就可以用extends

如果只是把抽象类当标准，我们就用implements

实现 implements

```
abstract class Animals
{
  void eat(){
    print("吃");
  }
}
```

```
class Dog implements Animals
{
  @override
  void eat(){
    print("吃骨头");
  }
}
```

☆单继承，多实现

混入 with (mixins)

```
作为混入的类只能继承自Object
没有构造函数
可以混入多个，如果有相同的方法，后者会替换前者
```

为了复用！！！

同时存在注意先后顺序 extends -> mixins -> implements

部分导入 show,hide

泛型，为了复用+校验

```
T getData<T>(T value){
  return value;
}
```

```
List list = new List<String>();
list.add(12);//错误的写法
list.add('12');//正确
```

内置库

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

外部库

```
https://pub.dev/packages
https://pub.dartlang.org/flutter
```

`pubspec.yaml`

```
  dependencies:
    http: ^0.12.0
```

自动或手动运行 pub get

自定义库

作业

实现一个抽象类Vehicle，定义数值车轮wheels，定义抽象方法getWheelsNum。实现一个汽车类继承自Vehicle。新建汽车实例调用getWheelsNum打印日志wheels数值。

暗号：你好Dartcolor