

07 | 函数、类与运算符：Dart是如何处理信息的？

2019-07-13 陈航

Flutter核心技术与实战

[进入课程 >](#)



讲述：陈航

时长 11:05 大小 10.17M



你好，我是陈航。

在上一篇文章中，我通过一个基本 hello word 的示例，带你体验了 Dart 的基础语法与类型变量，并与其他编程语言的特性进行对比，希望可以帮助你快速建立起对 Dart 的初步印象。

其实，编程语言虽然千差万别，但归根结底，它们的设计思想无非就是回答两个问题：

如何表示信息；

如何处理信息。

在上一篇文章中，我们已经解决了 Dart 如何表示信息的问题，今天这篇文章我就着重和你分享它是如何处理信息的。

作为一门真正面向对象的编程语言，Dart 将处理信息的过程抽象为了对象，以结构化的方式将功能分解，而函数、类与运算符就是抽象中最重要的手段。

接下来，我就从函数、类与运算符的角度，来进一步和你讲述 Dart 面向对象设计的基本思路。

函数


函数是一段用来独立地完成某个功能的代码。我在上一篇文章中和你提到，在 Dart 中，所有类型都是对象类型，函数也是对象，它的类型叫作 Function。这意味着函数也可以被定义为变量，甚至可以被定义为参数传递给另一个函数。

在下面这段代码示例中，我定义了一个判断整数是否为 0 的 isZero 函数，并把它传递给了另一个 printInfo 函数，完成格式化打印出判断结果的功能。

 复制代码

```
1 bool isZero(int number) { // 判断整数是否为 0
2   return number == 0;
3 }
4
5 void printInfo(int number,Function check) { // 用 check 函数来判断整数是否为 0
6   print("$number is Zero: ${check(number)}");
7 }
8
9 Function f = isZero;
10 int x = 10;
11 int y = 0;
12 printInfo(x,f); // 输出 10 is Zero: false
13 printInfo(y,f); // 输出 0 is Zero: true
```

如果函数体只有一行表达式，就比如上面示例中的 isZero 和 printInfo 函数，我们还可以像 JavaScript 语言那样用箭头函数来简化这个函数：

 复制代码

```
1 bool isZero(int number) => number == 0;
```

```
2
3 void printInfo(int number,Function check) => print("$number is Zero: ${check(number)}")
```

有时，一个函数中可能需要传递多个参数。那么，如何让这类函数的参数声明变得更加优雅、可维护，同时降低调用者的使用成本呢？

C++ 与 Java 的做法是，提供函数的重载，即提供同名但参数不同的函数。但**Dart 认为重载会导致混乱，因此从设计之初就不支持重载，而是提供了可选命名参数和可选参数。**

具体方式是，在声明函数时：

给参数增加{}，以 paramName: value 的方式指定调用参数，也就是可选命名参数；

给参数增加 []，则意味着这些参数是可以忽略的，也就是可选参数。

在使用这两种方式定义函数时，我们还可以在参数未传递时设置默认值。我以一个只有两个参数的简单函数为例，来和你说明这两种方式的具体用法：

 复制代码

```
1 // 要达到可选命名参数的用法，那就在定义函数的时候给参数加上 {}
2 void enable1Flags({bool bold, bool hidden}) => print("$bold , $hidden");
3
4 // 定义可选命名参数时增加默认值
5 void enable2Flags({bool bold = true, bool hidden = false}) => print("$bold , $hidden");
6
7 // 可忽略的参数在函数定义时用 [] 符号指定
8 void enable3Flags(bool bold, [bool hidden]) => print("$bold , $hidden");
9
10 // 定义可忽略参数时增加默认值
11 void enable4Flags(bool bold, [bool hidden = false]) => print("$bold , $hidden");
12
13 // 可选命名参数函数调用
14 enable1Flags(bold: true, hidden: false); //true, false
15 enable1Flags(bold: true); //true, null
16 enable2Flags(bold: false); //false, false
17
18 // 可忽略参数函数调用
19 enable3Flags(true, false); //true, false
20 enable3Flags(true,); //true, null
21 enable4Flags(true); //true, false
22 enable4Flags(true,true); // true, true
```

这里我要和你强调的是，在 Flutter 中会大量用到可选命名参数的方式，你一定要记住它的用法。

类

类是特定类型的数据和方法的集合，也是创建对象的模板。与其他语言一样，Dart 为类概念提供了内置支持。


类的定义及初始化

Dart 是面向对象的语言，每个对象都是一个类的实例，都继承自顶层类型 Object。在 Dart 中，实例变量与实例方法、类变量与类方法的声明与 Java 类似，我就不再过多展开了。

值得一提的是，Dart 中并没有 public、protected、private 这些关键字，我们只要在声明变量与方法时，在前面加上 “_” 即可作为 private 方法使用。如果不加 “_”，则默认为 public。不过，“_” 的限制范围并不是类访问级别的，而是库访问级别。

接下来，我们以一个具体的案例看看 Dart 是如何定义和使用类的。

我在 Point 类中，定义了两个成员变量 x 和 y，通过构造函数语法糖进行初始化，成员函数 printInfo 的作用是打印它们的信息；而类变量 factor，则在声明时就已经赋好了默认值 0，类函数 printZValue 会打印出它的信息。


 复制代码

```
1 class Point {
2   num x, y;
3   static num factor = 0;
4   // 语法糖，等同于在函数体内: this.x = x;this.y = y;
5   Point(this.x,this.y);
6   void printInfo() => print('$x, $y');
7   static void printZValue() => print('$factor');
8 }
9
10 var p = new Point(100,200); // new 关键字可以省略
11 p.printInfo(); // 输出 (100, 200);
12 Point.factor = 10;
13 Point.printZValue(); // 输出 10
```

有时候类的实例化需要根据参数提供多种初始化方式。除了可选命名参数和可选参数之外，Dart 还提供了**命名构造函数**的方式，使得类的实例化过程语义更清晰。

此外，与 C++ 类似，Dart 支持**初始化列表**。在构造函数的函数体真正执行之前，你还有机会给实例变量赋值，甚至重定向至另一个构造函数。

如下面实例所示，Point 类中有两个构造函数 Point.bottom 与 Point，其中：Point.bottom 将其成员变量的初始化重定向到了 Point 中，而 Point 则在初始化列表中为 z 赋上了默认值 0。

 复制代码

```
1 class Point {
2   num x, y, z;
3   Point(this.x, this.y) : z = 0; // 初始化变量 z
4   Point.bottom(num x) : this(x, 0); // 重定向构造函数
5   void printInfo() => print('$x,$y,$z');
6 }
7
8 var p = Point.bottom(100);
9 p.printInfo(); // 输出 (100,0,0)
```

复用

在面向对象的编程语言中，将其他类的变量与方法纳入本类中进行复用的方式一般有两种：**继承父类和接口实现**。当然，在 Dart 也不例外。


在 Dart 中，你可以对同一个父类进行继承或接口实现：

继承父类意味着，子类由父类派生，会自动获取父类的成员变量和方法实现，子类可以根据需要覆写构造函数及父类方法；

接口实现则意味着，子类获取到的仅仅是接口的成员变量符号和方法符号，需要重新实现成员变量，以及方法的声明和初始化，否则编译器会报错。

接下来，我以一个例子和你说明在 Dart 中**继承和接口的差别**。

Vector 通过继承 Point 的方式增加了成员变量，并覆写了 printInfo 的实现；而 Coordinate，则通过接口实现的方式，覆写了 Point 的变量定义及函数实现：

 复制代码

```
1 class Point {
2     num x = 0, y = 0;
3     void printInfo() => print('$x,$y');
4 }
5
6 //Vector 继承自 Point
7 class Vector extends Point{
8     num z = 0;
9     @override
10    void printInfo() => print('$x,$y,$z'); // 覆写了 printInfo 实现
11 }
12
13 //Coordinate 是对 Point 的接口实现
14 class Coordinate implements Point {
15     num x = 0, y = 0; // 成员变量需要重新声明
16     void printInfo() => print('$x,$y'); // 成员函数需要重新声明实现
17 }
18
19 var xxx = Vector();
20 xxx
21   ..x = 1
22   ..y = 2
23   ..z = 3; // 级联运算符，等同于 xxx.x=1; xxx.y=2;xxx.z=3;
24 xxx.printInfo(); // 输出 (1,2,3)
25
26 var yyy = Coordinate();
27 yyy
28   ..x = 1
29   ..y = 2; // 级联运算符，等同于 yyy.x=1; yyy.y=2;
30 yyy.printInfo(); // 输出 (1,2)
31 print (yyy is Point); //true
32 print(yyy is Coordinate); //true
```


可以看出，子类 Coordinate 采用接口实现的方式，仅仅是获取到了父类 Point 的一个“空壳子”，只能从语义层面当成接口 Point 来用，但并不能复用 Point 的原有实现。那么，我们是否能够找到方法去复用 Point 的对应方法实现呢？

也许你很快就想到了，我可以让 Coordinate 继承 Point，来复用其对应的方法。但，如果 Coordinate 还有其他的父类，我们又该如何处理呢？

其实，除了继承和接口实现之外，Dart 还提供了另一种机制来实现类的复用，即“混入”（Mixin）。混入鼓励代码重用，可以被视为具有实现方法的接口。这样一来，不仅可以解决 Dart 缺少对多重继承的支持问题，还能够避免由于多重继承可能导致的歧义（菱形问题）。

备注：继承歧义，也叫菱形问题，是支持多继承的编程语言中一个相当棘手的问题。当 B 类和 C 类继承自 A 类，而 D 类继承自 B 类和 C 类时会产生歧义。如果 A 中有一个方法在 B 和 C 中已经覆写，而 D 没有覆写它，那么 D 继承的方法的版本是 B 类，还是 C 类的呢？

要使用混入，只需要 with 关键字即可。我们来试着改造 Coordinate 的实现，把类中的变量声明和函数实现全部删掉：

 复制代码

```
1 class Coordinate with Point {  
2 }  
3  
4 var yyy = Coordinate();  
5 print (yyy is Point); //true  
6 print(yyy is Coordinate); //true
```

可以看到，通过混入，一个类里可以以非继承的方式使用其他类中的变量与方法，效果正如你想象的那样。

运算符

Dart 和绝大部分编程语言的运算符一样，所以你可以用熟悉的方式去执行程序代码运算。不过，Dart 多了几个额外的运算符，用于简化处理变量实例缺失（即 null）的情况。

?.运算符：假设 Point 类有 printInfo() 方法，p 是 Point 的一个可能为 null 的实例。那么，p 调用成员方法的安全代码，可以简化为 p?.printInfo()，表示 p 为 null 的时候跳过，避免抛出异常。

??= 运算符：如果 a 为 null，则给 a 赋值 value，否则跳过。这种用默认值兜底的赋值语句在 Dart 中我们可以用 a ??= value 表示。


??运算符：如果 a 不为 null，返回 a 的值，否则返回 b。在 Java 或者 C++ 中，我们需要通过三元表达式 (a != null)? a : b 来实现这种情况。而在 Dart 中，这类代码可以简化为 a ?? b。

在 Dart 中，一切都是对象，就连运算符也是对象成员函数的一部分。

对于系统的运算符，一般情况下只支持基本数据类型和标准库中提供的类型。而对于用户自定义的类，如果想支持基本操作，比如比较大小、相加相减等，则需要用户自己来定义关于这个运算符的具体实现。

Dart 提供了类似 C++ 的运算符覆写机制，使得我们不仅可以覆写方法，还可以覆写或者自定义运算符。

接下来，我们一起看一个 Vector 类中自定义 “+” 运算符和覆写 “==” 运算符的例子：

 复制代码

```
1 class Vector {
2   num x, y;
3   Vector(this.x, this.y);
4   // 自定义相加运算符，实现向量相加
5   Vector operator +(Vector v) => Vector(x + v.x, y + v.y);
6   // 覆写相等运算符，判断向量相等
7   bool operator == (dynamic v) => x == v.x && y == v.y;
8 }
9
10 final x = Vector(3, 3);
11 final y = Vector(2, 2);
12 final z = Vector(1, 1);
13 print(x == (y + z)); // 输出 true
14
```

operator 是 Dart 的关键字，与运算符一起使用，表示一个类成员运算符函数。在理解时，我们应该把 operator 和运算符作为整体，看作是一个成员函数名。

总结

函数、类与运算符是 Dart 处理信息的抽象手段。从今天的学习中你可以发现，Dart 面向对象的设计吸纳了其他编程语言的优点，表达和处理信息的方式既简单又简洁，但又不失强

大。

通过这两篇文章的内容，相信你已经了解了 Dart 的基本设计思路，熟悉了在 Flutter 开发中常用的语法特性，也已经具备了快速上手实践的能力。

接下来，我们简单回顾一下今天的内容，以便加深记忆与理解。

首先，我们认识了函数。函数也是对象，可以被定义为变量，或者参数。Dart 不支持函数重载，但提供了可选命名参数和可选参数的方式，从而解决了函数声明时需要传递多个参数的可维护性。


然后，我带你学习了类。类提供了数据和函数的抽象复用能力，可以通过继承（父类继承，接口实现）和非继承（Mixin）方式实现复用。在类的内部，关于成员变量，Dart 提供了包括命名构造函数和初始化列表在内的两种初始化方式。

最后，需要注意的是，运算符也是对象成员函数的一部分，可以覆写或者自定义。

思考题

最后，请你思考以下两个问题。

1. 你是怎样理解父类继承，接口实现和混入的？我们应该在什么场景下使用它们？
2. 在父类继承的场景中，父类子类之间的构造函数执行顺序是怎样的？如果父类有多个构造函数，子类也有多个构造函数，如何从代码层面确保父类子类之间构造函数的正确调用？

 复制代码

```
1 class Point {
2   num x, y;
3   Point() : this.make(0,0);
4   Point.left(x) : this.make(x,0);
5   Point.right(y) : this.make(0,y);
6   Point.make(this.x, this.y);
7   void printInfo() => print('$x,$y');
8 }
9
10 class Vector extends Point{
11   num z = 0;
12   /*5 个构造函数
13   Vector
```

```
14 Vector.left;
15 Vector.middle
16 Vector.right
17 Vector.make
18 */
19 @override
20 void printInfo() => print('$x,$y,$z'); // 覆写了 printInfo 实现
21 }
```

欢迎将你的答案留言告诉我，我们一起讨论。感谢你的收听，也欢迎你把这篇文章分享给更多的朋友一起阅读。



Flutter 核心技术与实战

来自 Google 的高性能跨平台开发框架

陈航

美团点评高级技术专家



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 06 | 基础语法与类型变量：Dart是如何表示信息的？

精选留言 (5)

写留言



Young

2019-07-13

1.一般来讲，单继承，多实现，混入是多继承

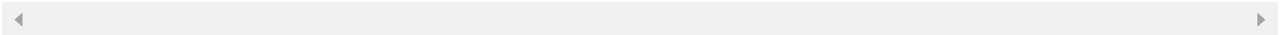
A.继承是子类需要复用父类的方法实现

B.实现接口是复用接口的参数，返回值，和方法名，但不复用方法的实现，在Dart中实现抽象类 更像在java中实现用interface修饰的接口

C.混入是多继承，当被混入的类有多个同名方法时，调用子类的该方法时，会调用with声...

展开 ∨

作者回复: 厉害了



💬 1

👍 7



兔子先生

2019-07-14

小白问下

Point(this.x, this.y) : z = 0;

这里的 ':' 的用法解释？ 和 '=>' 有什么区别？

展开 ∨

💬 2

👍



Phony Lou

2019-07-14

一个小问题，在覆写相等运算符时为何需要传入dynamic变量，而不能传入Vector呢？

bool operator==(dynamic v) => x == v.x && y == v.y;

bool operator==(Vector v) => x == v.x && y == v.y; // 报错

展开 ∨

💬

👍



宋锡珺

2019-07-14

1.

父类继承：和java类似，继承了父类的实例变量和各种方法。但是不能用一个普通方法重写getter。

抽象类：抽象类不能实例化，会报出AbstractClassInstantiationError错误。

接口：成员变量，成员函数需要重新声明实现。和java不一样的是，没有接口声明，可以...

展开 ∨

💬

👍



Mkl

2019-07-13

老师您好，请问“我们只要在声明变量与方法时，在前面加上 `private` 即可作为 `private` 方法使用。如果不加 `private`，则默认为 `public`。”这里边的双引号中的内容是什么，看不到呀😭

作者回复: 是 `private`

感谢提醒，我们改一下

