> 想学Flutter，就请关注这个专栏 Flutter系列（一）——详细介绍
> Flutter系列（二）——与React Native进行对比
> Flutter系列（三）——环境搭建（Windows）
> Flutter系列（四）——HelloWorld
> Dart语言详解（一）——详细介绍
> Dart语言详解（二）——基本语法

文档归档： https://github.com/yang0range/flutterfile

# 前言

之前，详细的介绍了一下Dart语言的前世今生，接下来，我们就该深入的了解Dart语言，了解Dart语言的基本数据类型，语法等相关的内容

# 变量和数据类型

## 变量

#### 命名

在Dart语言当中，定义变量的方式是 `dataType viriableName = Initial Valute` 这里其实和Java是一样的，不仅如此连命名的方式也和Java相同，都是采用驼峰命名的方式。

#### 声明

在声明的时候，除了基本数据类型之外，还有三种变量的声明。

**var，dynamic，Object**

**var**：如果没有初始值的时候，var可以变成任意类型。

**dynamic**：动态任意类型，编译阶段不检查类型。

**Object**： 动态任意类型，在编译阶段检查类型。

和var的区别： **var如果有初始值，那么类型会被锁定。**

#### 变量的默认值

1.没有初始值的变量会自动获取一个默认值**null**。
2.一切皆为对象，对象的默认值是**null**。

#### final和const

表示不可改变
相同点： - 声明的类型可以省略 - 初始化后不能再赋值 - 不能和var同时使用

不同点：

- 类级别常量，使用static const
- const可使用其他const 常量的值来初始化其值
- 使用const赋值声明，const可以省略
- 可以更改非final、非const变量的值。即使曾经具有const值
- const导致的不可变性是可以传递的
- 相同的const常量不会再内存中重复创建
- const需要是编译时常量

## 基本数据类型

Dart总共又8种基本数据类型。 分别是：

#### Numbers 数值型

数值型里面有包含了int型和doubule两种类型，这两种和Java类似，其中int是整数型，double是64-bit双精度浮点数，这两种都是Numbers类型的子类。

#### String

Dart字符串是UTF-16编码的字符序列。

1.可以使用单引号或者双引号来创建字符
例如: `void main() { print("This is a String"); print('This is also a stirng'); }` 这两个是一样的。

2.字符串拼接
例如： ``` void main() { String s1 = "First string."; String s2 = "Second string"; print(s1 + s2); }

```

3.字符插值 **${exprsssion}**，如果表达式是一个标识符，可以省略{}，如果表达式的结果为一个对象，Dart会调用对象的toString()函数来获取一个字符串。
例如： ``` void main() { String s1 = "First string."; print("String $s1"); }

void main() { print("The sum of 1 and 1 equals ${1 + 1}."); }

```

4.多行显示
使用双引号创建多行字符 还可以使用(')和(") 例如： `var s = 'Firtst' 'Second' "Third"; print(s);`

使用三引表示多行字符
例如 `var multilineString = """This is a multiline string consistiong of multiple lines"""; print(multilineString);`

5.使用r前缀创建"原始raw"字符串 ``` String s4 ="adbchi\ndfafa"; print(s4);

String s5 =r"adbchi\ndfafa"; print(s5); 输出的结果 I/flutter (17682): adbchi I/flutter (17682): dfafa I/flutter (17682): adbchi\ndfafa ```

#### Bool

Dart的bool和Java类似只有两种类型，一种是**true**一种是**false**，但是，不同的是bool对象未初始化的默认值是null。

#### List

- 因为在Dart当中，由一切皆为对象的概念，Dart就可以直接打印list包含list的元素，这点和Java由明显的不同，java中直接打印list结果是地址值。
  例如 `var list = [0, 1, 2, 3, 4, 5, 6]; print(list);` 打印出结果 `I/flutter (24300): [0, 1, 2, 3, 4, 5, 6]`
- Dart中List的下标索引和java一样都是从0开始。
- Dart中List也支持泛型，这点和java一样，同时还可以进行泛型的判断。 ``` var list1 = List(); print(list1 is List); 输出： true

- 有增删改查的操作，支持倒叙，自带顺序，洗牌，可以使用+将两个List合并。 例如 var list = [0, 1, 2, 3, 4, 5, 6]; print(list); var list2 = [7,8,9,10,11];

print(list+list2); 输出： [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] ```

#### Set

Dart里面的Set和Java类似，也是不能存放重复的元素。 - 两种初始化方式
Set有两种初始化方式 `var setName = <dataType>{} Set<dataType> setName ={}` - 大部分操作是和Java类似的
**不同的地方：** - difference：返回set1集合里面有但是set2里面没有的元素集合
`Set<String> set1 = {"1","2","3","4","5"}; Set<String> set2 = {"1","2","3"}; print(set1.difference(set2)); 输出： {4, 5}` - intersection：返回set1和set2的交集

`Set<String> set1 = {"1","2","3","4","5"}; Set<String> set2 = {"1","2","3"}; print(set1.intersection(set2)); 输出： {1, 2, 3}` -
union：返回set1和set2的并集
`Set<String> set1 = {"1","2","3","4","5"}; Set<String> set2 = {"1","2","3"}; print(set1.union(set2)); 输出： {1, 2, 3, 4, 5}` -
retainAll:set1只保留某些元素（要保留的元素要在原set中存在）

#### Map

和Java类似

#### Runes

Runes再Dart当中是字符的UTF-32的编码 ``` Main(){ Runes runes = new Runes('\u{1f605} \u6211'); var str1 = String.fromCharCodes(runes); print(str1); }

```

#### Symbol

Symbol标识符，主要是反射的作用，现在再mirrors模块已经被移除了。

### 函数

#### 定义

总体来说，和java类似。
其他几个特单： - 可在函数内定义 - 定义函数时可以省略类型 - 支持缩写语法=>

```
int add(int a, int b) => a + b;
```

#### 可选参数

- 可以选择命名参数
- 可以选择位置参数
- 可以添加默认参数 ``` printer(num n,{String s1, String s2}) {  print(n);  print(s1);  print(s2); }

main() {  printer(75, s1: 'hello'); } ```

``` printer(num n,{String s1, String s2}) {  print(n);  print(s1);  print(s2); }

main() {  printer(75); } ```

``` printer(num n,{String s1, String s2}) {  print(n);  print(s1);  print(s2); }

main() {  printer(75, s1: 'hello', s2: 'there'); } ```

``` String mysteryMessage(String who, [String what, String where]){  var message = '$who';  if(what != null && where == null){
  message = '$message said $what';  } else if (where != null){  message = '$message said $what at $where';  }  return message; }

main() {  var result = mysteryMessage('Billy', 'howdy', 'the ranch');  print(result); } ```

#### 匿名函数

- 可以赋值给变量，通过变量调用
- 可以在其他函数当中直接调用或者传递给其他函数 ``` //匿名函数 //赋值给变量 //无参匿名函数 var anonFunc1 = () => print('无参匿名函数');
  anonFunc1();

//有参匿名函数 var anonFunc = (name) => 'I am $name'; print(anonFunc('damon'));

//通过()调用，不推荐 // (()=>print('不推荐'))();

//匿名函数传参 List test(List list, String func(str)) { for (var i = 0; i < list.length; i++) { list[i] = func(list[i]); } return list; }

var list = ['d', 'a', 'm', 'o', 'n']; print(test(list, (str) => str * 2)); //String * int, Dart和Python可以这样用

//List.forEach()就用的匿名函数 List list1 = [11, 12, 13]; list1.forEach((item) => print('$item'));

//返回Function对象（闭包） Function makeAddFunc(int x) { x++; return (int y) => x + y; }

var addFunc = makeAddFunc(2); print(addFunc(3));

// 函数别名 MyFunc myFunc; //可以指向任何同签名的函数 myFunc = subtsract; myFunc(4, 2); myFunc = divide; myFunc(4, 2); //typedef 作为参数传递给函数 calculator(4, 2, subtsract); }

//函数别名 typedef MyFunc(int a, int b); //根据MyFunc相同的函数签名定义两个函数 subtsract(int a, int b) { print('subtsract: ${a - b}'); }

divide(int a, int b) { print('divide: ${a / b}'); } //typedef 也可以作为参数传递给函数 calculator(int a, int b, MyFunc func) { func(a, b); } ```

### 逻辑运算

#### 操作符

操作符大部分也和Java相同。

标红的是不同的，下面着重介绍一下。

**?.**

条件成员访问 和 . 类似，但是左边的操作对象不能为 null，例如 foo?.bar 如果 foo 为 null 则返回 null，否则返回 bar 成员。 ``` String a; print(a?.length); 输出null

String a="aaa"; print(a?.length); 输出 3 ```

**~/**

取商操作符
被除数 ÷ 除数 = 商 ... 余数，A ~/ B = C，这个C就是商。相当于Java里的 /

**：as、is、is!**

类型判定操作
类型判定操作符：as、is、is!在运行时判定对象类型 ``` //as 类型转换 num iNum = 1; num dNum = 1.0; int i = iNum as int; double d = dNum as double; print([i, d]);

// String s = iNum as String;

//is 如果对象是指定的类型返回 True print(iNum is int); Child child; Child child1 = new Child(); print(child is Parent); //child is Null print(child1 is Parent);

//is! 如果对象是指定的类型返回 False print(iNum is! int); ```

**..**

级联操作符
连续调用多个函数以及访问成员变量。
两个点的级联语法不是一个操作符。 只是一个 Dart 特殊语法。
`StringBuffer sb = new StringBuffer(); sb ..write('dongnao') ..write('flutter') ..write('\n') ..writeln('damon');`

#### 流程控制语句

- if else
- for，forEach，for-in
- while，do-while

- break，continue
- switch case
- assert
  大部分使用方法都是和Java相同的。
  不同的有

**for-in**

```
var colorList = ['black','red','yellow']; for(var i in colorList){ print(i); } 输出： I/flutter (31601): black I/flutter (31601): red I/flutter (31601): yellow
```

**assert**

断言，它可以为代码执行设置条件，用于bool条件为false时中断正常的运行。
assert(condition,optional,message)

```
var variable; print(variable); assert(variable!=null); variable = 6; print(variable); 输出： [ERROR:flutter/lib/ui/ui_dart_state.cc(157)] Unhandled Exception: 'package:flutter_app/main.dart': Failed assertion: line 50 pos 10: 'variable!=null': is not true.
```

## 最后

这一篇文章，我们详细的介绍了一下Dart语言的基本语法，了解Java的同学，对这些应该不陌生。接下来一篇文章，会对Dart语言继续进行详细介绍，让大家对Dart有一个高阶，全面的认识。

**Flutter**已经是**Top20的软件库，通过接下来的一系列的文章，希望我和大家一起来学习Flutter**，一起进步，一起有所收获，掌握未来技术主流的主动权！

有什么好的建议，意见，想法欢迎给我留言！

### 欢迎关注公共号

### 关注公众号会有更多收获！





### 动动小手指点赞，收藏，转发一键三连走一波吧！