1、论数字、函数和null都是对象

在变量中可以放置的所有东西都是*对象*,而每个对象都是*类*的实例。无论数字、函数和null都是对象。所有对象都继承自[Object]类。

```
//    abstract class int extends num
int variable = 3 ;
 variable.toString()
 //  abstract class Function
    Function funcTest  = myFunc;
    myFunc.toString();
     null.toString();//null
     null.runtimeType ;//Null

 void myFunc(int count){

 }
```

2、dart类型安全

（暂时不理会类型安全的定义，尤其是访问未被授权的内存）

Dart的类型安全性意味着您不能使用if(非booleanvalue)或assert(非booleanvalue)之类的代码。相反，应该显式地检查值 if(tue)

3、关键字

abstract (1)	do	import (1)	super
as (1)	dynamic (1)	in	switch
assert	else	interface (1)	sync* (2)
async (2)	enum	is	this
async* (2)	export (1)	library (1)	throw
await (2)	external (1)	mixin (1)	true
break	extends	new	try
case	factory (1)	null	typedef (1)
catch	false	operator (1)	var
class	final	part (1)	void
const	finally	rethrow	while
continue	for	return	with
covariant (1)	get (1)	set (1)	yield (2)
default	if	static (1)	yield*
deferred (1)	implements (1)		

4、变量

- var name = 'Bob';
- dynamic name = 'Bob'
- Object va = 9;
- String name = 'Bob';

默认值

未初始化的变量的初始值为null。甚至具有数字类型的变量最初也是null，因为数字——就像dart中的其他东西一样——是对象。

```
int lineCount;
```

5、Final 和 const修饰符

Final 变量只能设置一次;(可以不再定义的时候设置值)

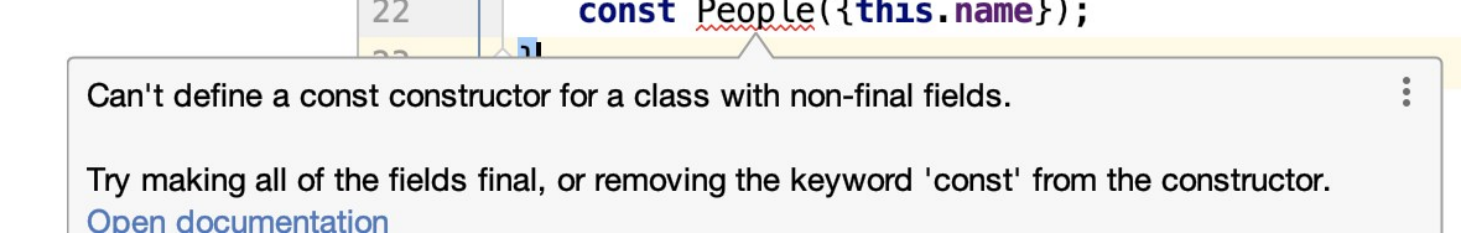
const变量是一个编译时常数。(Const变量是隐式最终变量。)最终的顶级或类变量在第一次使用时被初始化。

const关键字不只是声明常量变量。您还可以使用它来创建常量值，以及声明创建常量值的构造函数

常量构造函数

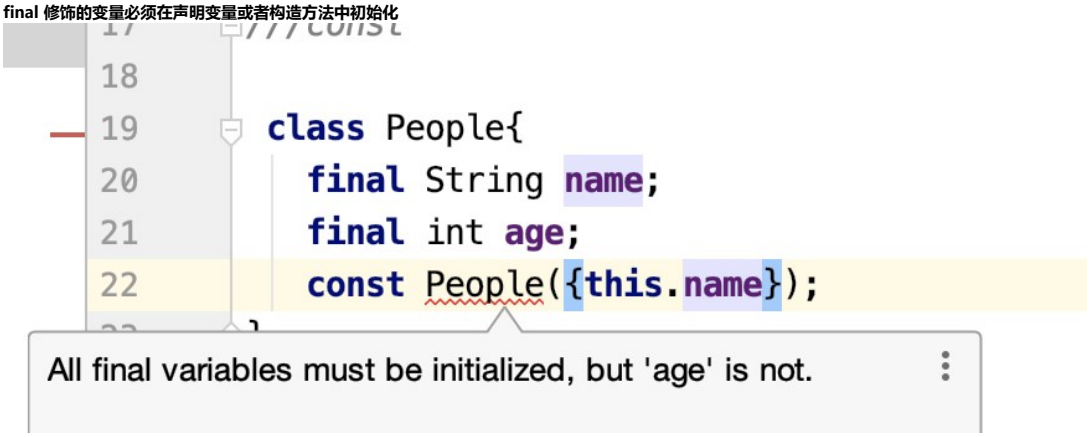
- 常量构造函数需以const关键字修饰
- const构造函数必须用于成员变量都是final的类

如下代码定义一个const对象，但是调用的构造方法不是const修饰的，则会报The constructor being called isn't a const constructor.错误



另外注意这个普通的构造函数也有的错误

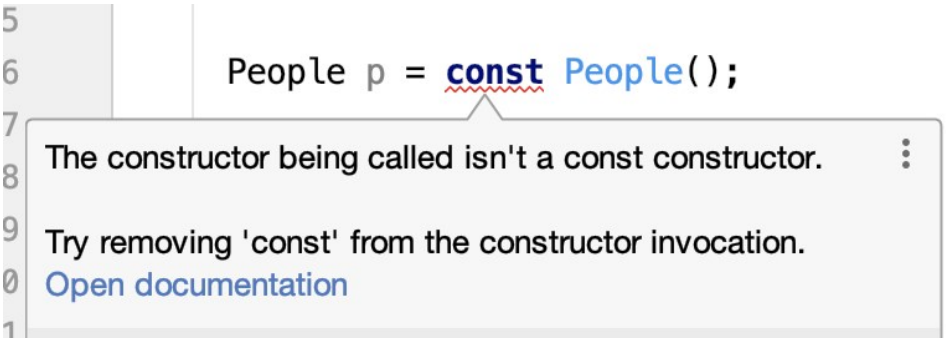
final 修饰的变量必须在声明变量或者构造方法中初始化

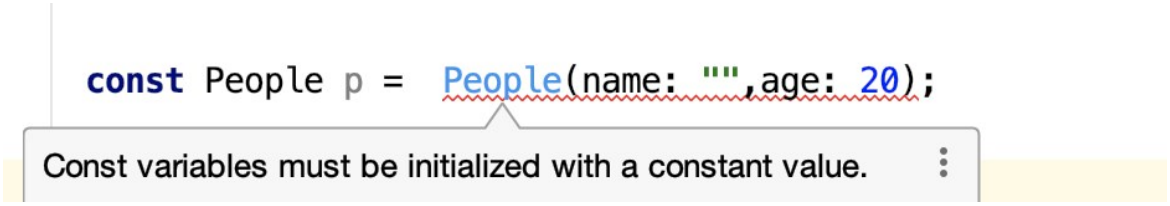


- 构建常量实例必须使用定义的常量构造函数

如下代码，定义一个const对象，但是调用的却是非常量构造函数，会报The constructor being called isn't a const constructor.错误

```
class People{
   final String name;
   final int age;
//   const People({this.name,this.age});

    People({this.name,this.age});
}
```





- 常量构造函数中的 修饰位置（= 左右的区别）

```
class People{
   final String name;
   final int age;
//   const People({this.name,this.age});

    People({this.name,this.age});
}

 People p =  People(name: "",age: 20);
p.age = 30 ;  //final 修改不可修改
```

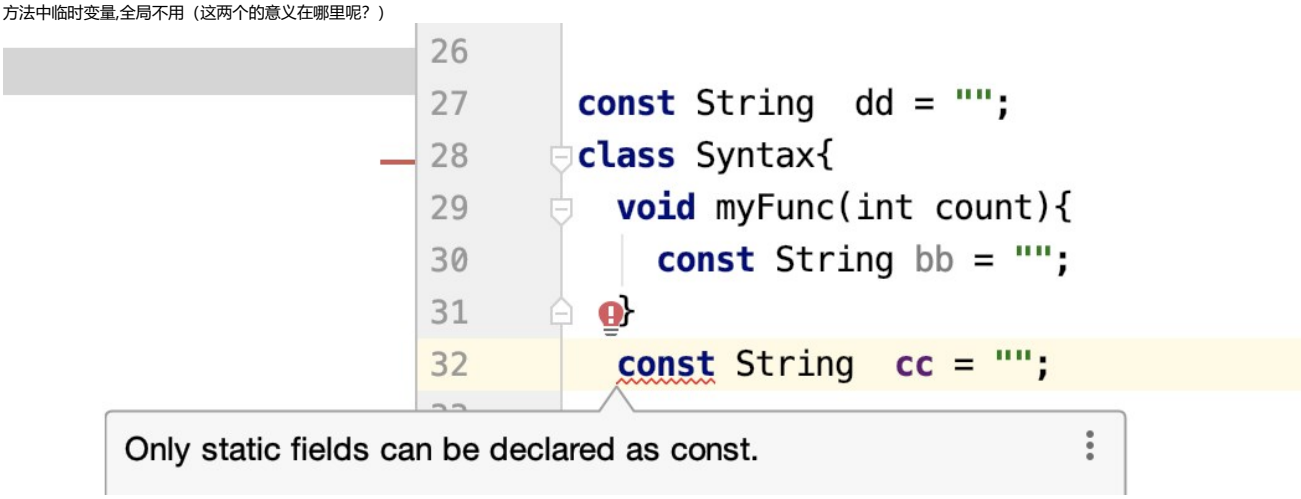
```
 class People{
  final String name;
  final int age;
  const People({this.name,this.age});
}
///下面这两个的区别
const  People p =  People(name: "",age: 20);
People p1 = const People(name: "",age: 20);

 //p = People(name: "1",age: 21);//报错，p 是常量不可修改
  p1 = const People(name: "1",age: 21);
```

```c
```

- const **修改变量**

作为成员变量必须使用static 编译时

方法中临时变量,全局不用（这两个的意义在哪里呢？）



6、函数

参数

- 普通参数
- 可选命名参数
- 可选位置参数
- 默认参数值

普通参数

```
void  say(String from,String msg) {
 }
 say("from", "msg");
```

可选命名参数

```
void  say(String from,String msg, {String device,int age}) {
  }
 say("from", "msg",device: "ios",age: 10);
```

可选位置参数

```
  void  say1(String from,String msg, [String device,int age]) {

  }
say1("from", "msg","device");
```

默认参数值（： = 似乎都可以）

```
void enableFlags({bool bold = false, bool hidden:false}) {
  print("$hidden");
}
```

注意：

> 可选参数(命名，位置)只能作为参数列表中的最后一项

> 可以使用 = 来定义命名和位置参数的默认值。默认值必须是编译时常量。如果没有提供默认值，则默认值为null。

可选参数可以是位置参数，也可以是命名参数，但不能两者都包含。

返回值

所有函数都会返回一个值。 如果没有明确指定返回值， 函数体会被隐式的添加 return null; 语句。

```
foo() {}

assert(foo() == null);
```

7、运算符

- ~/ 代表 整除（返回值为int）
- / 返回值 为double
- 上面两个的 赋值运算符 分别是 ~/= 和 /=
- ?? 代表 if null

> expr1 ?? expr2
> 如果expr1是非空的，则返回其值；否则，计算并返回expr2的值。

- ??=

> 仅仅在b为空的情况下b被赋值value否则b的值不变
> var b ??= value;

- is 判断是某一个类型

- is! 判断不是某一个类型

```
int variable = 3 ;
variable is int  // true
variable is String// false
variable is! int  false
variable is! String// true
```

- as
  - as操作符将对象转换为特定类型

```
if (emp is Person) {
  // Type check
  emp.firstName = 'Bob';
}
```

使用as操作符可以使代码更简短:

```
(emp as Person).firstName = 'Bob';
```

> 注意:代码不等效。如果emp是null或不是Person，那么第一个示例(使用is)什么都不做;第二个(带有as)抛出异常。

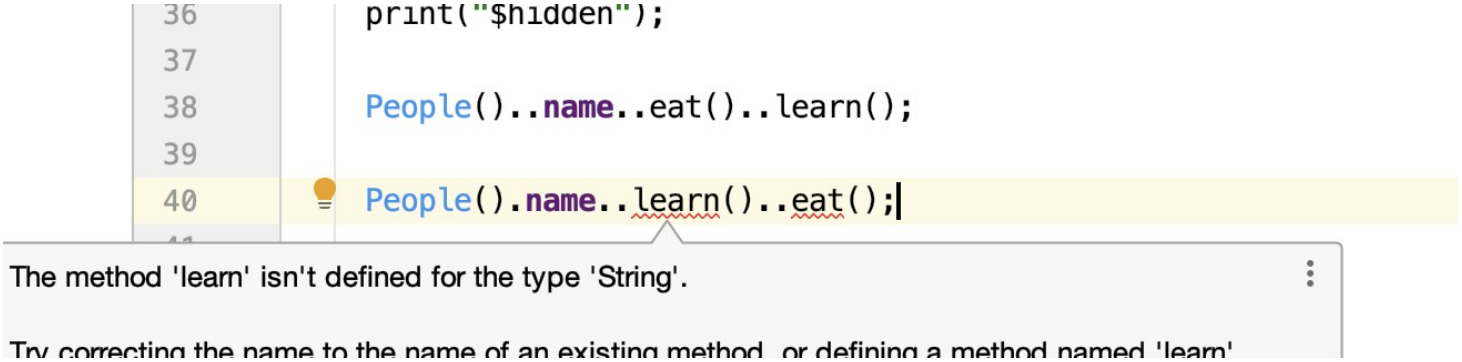
  - 指定一个库前缀

```
import 'package:rhflutterapp/utils/screen_util.dart' as KScreen;
 KScreen.ScreenUtil.sharedInstance()
```

- .. 级联表示法

> 注意:严格地说，级联的“..”表示法不是运算符。这只是Dart语法的一部分。

```
class People{
   String name;
   void eat(){}
   int learn(){}
}
```



- . 和 ?.

```
p?.name;
p?.eat();
```

判null 为null 后面的不执行 和swift一样

如果p为null ，使用（?.）如果取值对象为空时返回null 但是使用（.）如果对象为空则抛出异常

分类: flutter

标签: flutter

肖无情
粉丝 - 2 关注 - 5
+加关注

« 上一篇： 01-01 iOS内存对齐、内存对齐算法
» 下一篇： 02-02dart语法

编辑推荐：
· gRPC 入门与实操（.NET 篇）
· dotnet 代码优化 聊聊逻辑圈复杂度
· 一个棘手的生产问题，但是我写出来之后，就是你的了
· 你可能不知道的容器镜像安全实践
· .Net 6 使用 Consul 实现服务注册与发现

阅读排行：
· Redux与前端表格施展“组合拳”，实现大屏展示应用的交互增强
· gRPC入门与实操(.NET篇)
· 博客园主题修改分享 - 过年篇
· 如何优雅地校验后端接口数据，不做前端背锅侠
· 产品与研发相处之道