02-04 flutter 构造函数总结

Posted on 2020-11-23 23:30 肖无情 阅读(1174) 评论(0) 编辑 收藏 举报

总结：

记住构造函数是不能被继承的，这将意味着子类不能继承父类的命名式构造函数，如果你想在子类中提供一个与父类命名构造函数名字一样的命名构造函数，则需要在子类中显式地声明

如果类A 没有显示声明构造函数，那么它将有一个默认的构造函数，这个构造函数 **没有参数**

如果这个类有父类，

a: 父类没有显示声明构造函数，或者写了一个普通无参构造

那么A构造函数（无论默认构造还是命名构造等），还会默认调用父类的无参数构造函数

b:父类显示声明构造函数

那么A类必须要有非默认构造函数（否则会把报错），且必须显示调用父类其中一个构造函数

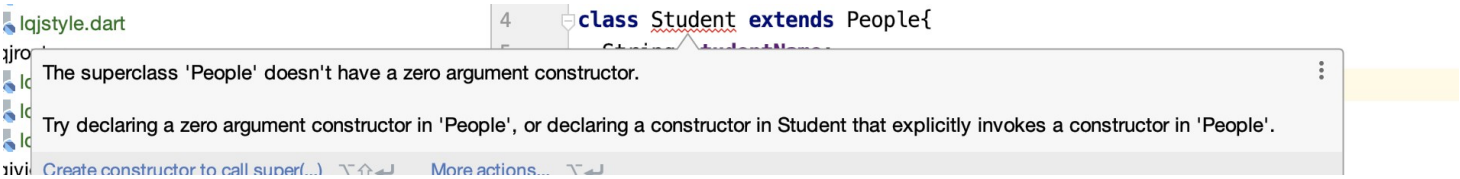
1、默认构造函数

如果你定义了一个类，而没有定义构造函数，那么它将有一个默认的构造函数，这个构造函数 **没有参数**

如果这个类有父类，那么默认构造函数，还会默认调用父类的无参数构造函数。

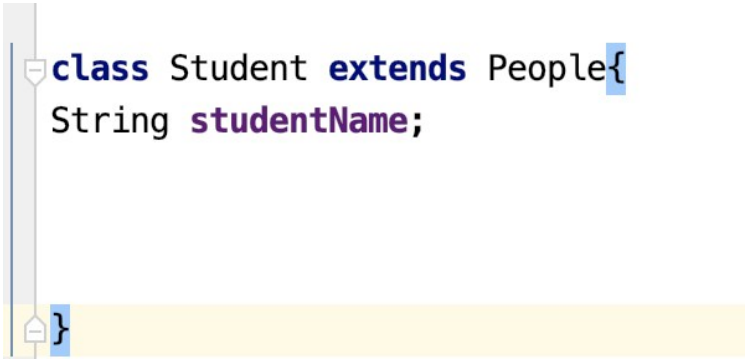
如果自己写了构造函数，那就会覆盖默认构造（这个不存在了），

```
class People{
  String peopleName;
  People(String peopleName){
  }
}

class Student extends People{
  String studentName;
}
```



自己可以写个同样的无参构造进行覆盖默认无参构造

```
class People{
  String name;
  int age;
  int sex;
  People(){

  }
}
```

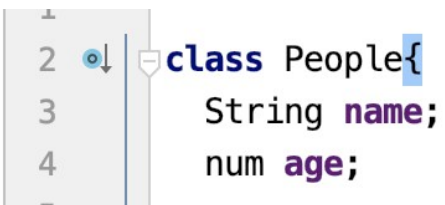


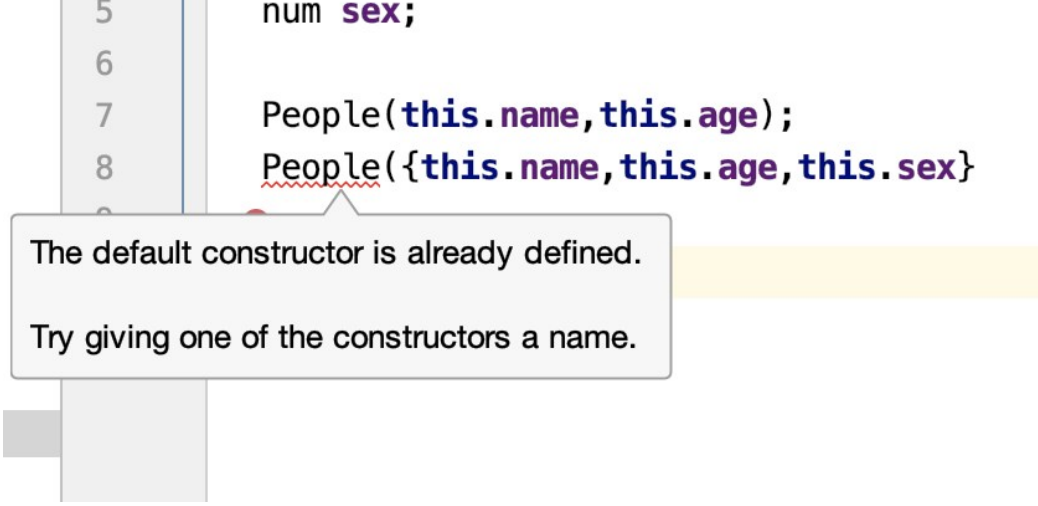
2、普通构造函数

声明一个和类名相同的函数，来作为类的构造函数

```
class Person {
    String name;
    num age;
    Person(String name, num age) {
      this.name = name;
      this.age = age;
    }
  }
```

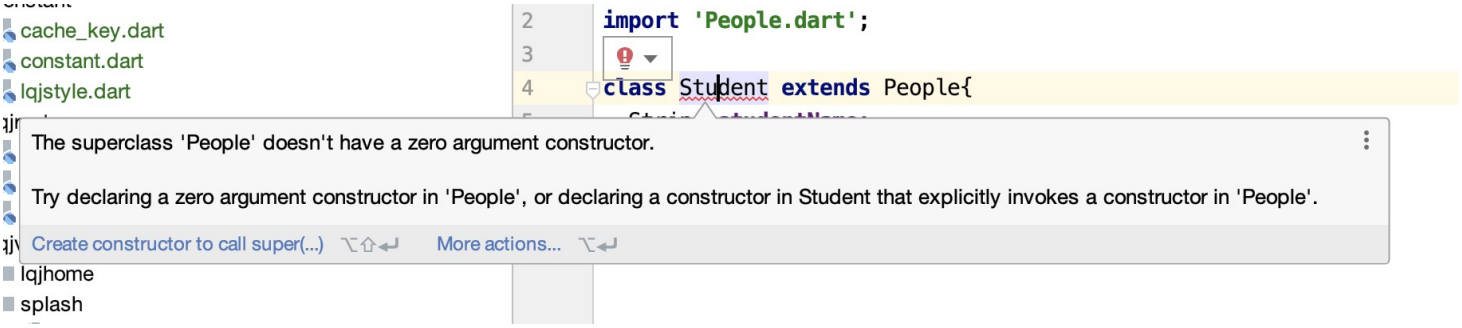
Dart 中可以简写：





说明 普通构造函数不能重载，

并且子类必须显示的调用



explicitly invokes a constructor 显示的调用一个构造函数

3、命名构造函数

使用命名构造函数可为一个类实现多个构造函数,但是同样不能重载

子类必须显式的调用其中的一个

```
class Student extends People{
  String studentName;

  Student.studentinit() : super.init();
}
```

这里可以实现 构造函数的私有化 比如单例中的使用

4、调用父类构造函数

默认情况下，子类的构造函数会调用父类的匿名无参数构造方法，并且该调用会在子类构造函数的函数体代码执行前，如果子类构造函数还有一个初始化列表，那么该初始化列表会在调用父类的该构造函数之前被执行，总的来说，这三者的调用顺序如下：

1. 初始化列表
2. 父类的构造函数
3. 当前类的构造函数

如果父类没有匿名无参数构造函数，那么子类必须调用父类的其中一个构造函数，为子类的构造函数指定一个父类的构造函数只需在构造函数体前使用（:）指定。

因为参数会在子类构造函数被执行前传递给父类的构造函数，因此该参数也可以是一个表达式，比如一个函数：

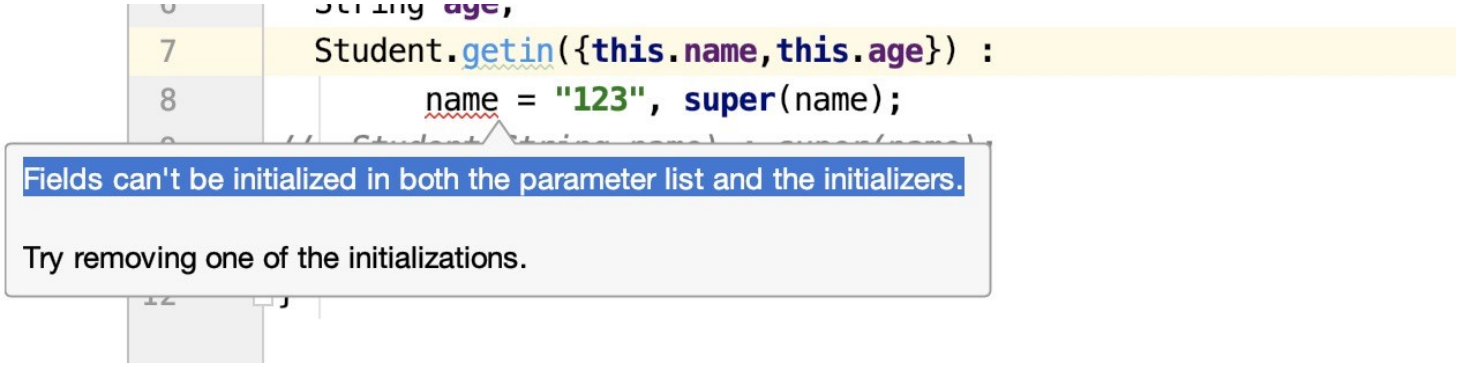
```
class Employee extends Person {
  Employee() : super.fromJson(defaultData);
```

```
  // ···
}
```

注意：

传递给父类构造函数的参数不能使用 this 关键字，因为在参数传递的这一步骤，子类构造函数尚未执行，子类的实例对象也就还未初始化，因此所有的实例成员都不能被访问，但是类成员可以。

```
class Student extends People{
String studentName;
String studentAge;
String studentSex;
int age1
Student(
    this.studentAge,
    {
      this.studentSex,
      int age
    }
    ):studentName = "1", this.age1 = age ?? 10,  super(age);

}
```



不能同时在构造器和初始化列表中同时初始化，

下面是两种不通过的写法

```
class People{
  String name;
  int age;
  int sex;
  People(int age){

  }

}

import 'People.dart';

class Student extends People {
  String studentName;
  String studentAge;
  String studentSex;
  String studentSex1;

  Student(this.studentAge,
      {
        this.studentSex,
        int age
      })
      :studentName = "1",
        super(age);


  Student.init(this.studentAge, {this.studentName: "22", this.studentSex1 = "33", int age})
      : studentSex = "1",
        super(age){

  }
}
```

5. 构造函数传递（重定向构造函数）

定义构造函数的时候，除了一个普通构造函数，还可以有若干命名构造函数，这些构造函数之间，有时候会有一些相同的逻辑，如果分别书写在各个构造函数中，会有些多余，所以构造函数可以传递。

```
class Point {
  num x, y;

  // The main constructor for this class.
  Point(this.x, this.y);

  // Delegates to the main constructor.
  Point.alongXAxis(num x) : this(x, 0);
}

class Point {
  num x, y,z;

  // The main constructor for this class.
  Point(this.x, this.y);
  Point.aaa(this.x,this.y,this.z);

  // Delegates to the main constructor.
  Point.alongXAxis(num x):this.aaa(x,0,x);
```

```
}
```

复制代码

传递构造函数，没有方法体，会在初始化列表中，调用另一个构造函数。

6常量构造函数

```
class ImmutablePoint {
  static final ImmutablePoint origin =
      const ImmutablePoint(0, 0);

  final num x, y;

  const ImmutablePoint(this.x, this.y);
}
```

复制代码

如果你的类，创建的对象永远不会改变，你可以在编译期就创建这个常量实例，并且定义一个常量构造函数，并且确保所有的成员变量都是final的。

7工厂构造函数的定义

工厂构造函数是一种构造函数,与普通构造函数不同,工厂函数不会自动生成实例,而是通过代码来决定返回的实例对象.

使用 factory 关键字标识类的构造函数将会令该构造函数变为工厂构造函数，这将意味着使用该构造函数构造类的实例时并非总是会返回新的实例对象。例如，工厂构造函数可能会从缓存中返回一个实例，或者返回一个子类型的实例。

以下示例演示了从缓存中返回对象的工厂构造函数:

```
class Logger {
  final String name;
  bool mute = false;

  // _cache 变量是库私有的, 因为在其名字前面有下划线。
  static final Map<String, Logger> _cache =
      <String, Logger>{};

  factory Logger(String name) {
    return _cache.putIfAbsent(
        name, () => Logger._internal(name));
  }

  Logger._internal(this.name);

  void log(String msg) {
    if (!mute) print(msg);
  }
}
```

在工厂构造函数中无法访问 this。

分类: flutter

标签: flutter

肖无情
粉丝 - 2 关注 - 5

« 上一篇： 02-03 flutter异步
» 下一篇： 02-05 flutter provider的使用