1. Null & null

Null是一个Trait，不能创建它的实例，但是Scala在语言层面上存在一个Null的实例，那就是null。Scala定义了一个类似于对象语义的Null，和一个值语义的null。

<https://www.cnblogs.com/moonandstar08/p/5759137.html>

1. Nil

Nil是一个继承List[Nothing]的对象，它就是一个空的列表。Nil就是一个可以封装任何东西的空容器。它的长度为0。它并不是一无所有，它是一个容器，一个列表，只是没有存放内容而已。

1. Nothing

Nothing是所有类型的子类，也是Null的子类。Nothing没有对象，但是可以用来定义类型。

例如，如果一个方法抛出异常，则异常的返回值类型就是Nothing(虽然不会返回)

1. Option

写Java程序的时候，经常会碰到没有有意义的东西可以返回，我们返回null。但返回null有一些问题，调用方必须检查返回值，不然会有NullPointerException的异常。这逼迫我们去check函数的返回值。还有一种解决办法是使用异常，但增加try/catch块，并不是明智的选择。

Scala内置一种解决办法。如果你想返回一个String，但可能有的时候得不到有意义的返回值，我们可以让函数返回Option[String]。

1. Unit

Unit跟java的void一样，表示函数没有返回值。

Unit类型是值类型，全局只存在唯一的值：()

scala> val a:Unit = ()

a: Unit = ()

若尝试把其它类型的值赋给它，也ok：

scala> val a:Unit = 2

a: Unit = () // 暂忽略编译器的警告，跟我们要关注的无关

诊断一下：

$ scala -nocompdaemon -Xprint:typer -e 'val a:Unit = 2'

…

private[this] val a: Unit = {

2;

() // 编译器自动增加了一个Unit的值：()

};

…

原因是编译对待等号右边的表达式，看待成为“行为”，会把这段“行为”统一放入花括号里。

并判断最后一句表达式的值是不是Unit类型的，如果不是则自动增加一个值：()

所以等号右边的 2 最终被转化为 { 2; () } 所以 foo(2,3,"OK") 也是被翻译为了

foo( {(2, 3, "OK"); ()} )

参数封装在一个tuple里，在最后返回一个 ()

小结：通常Unit只用来声明函数或方法的返回值，其他场景基本是没有意义的。