如果你熟悉 Java 的话，应该知道 Java 中的 switch 语句并不怎么好用。首先，switch 只能传入整型或短于整型的变量作为条件，JDK 1.7 之后增加了对字符串变量的支持，但如果你的判断逻辑使用的并非是上述几种类型的变量，那么不好意思，switch 并不适合你。其次，switch 中的每个 case 条件都要在最后主动加上一个 break，否则执行完当前 case 之后会依次执行下面的 case，这一特性曾经导致过无数诡异的 bug，就是因为有人忘记添加 break。

而 Kotlin 中的 when 语句不仅解决了上述痛点，还增加了许多更为强大的新特性，有时候它比 if 语句还要简单好用，现在我们就来学习一下吧。

我准备带你编写一个查询考试成绩的功能，输入一个学生的姓名，返回该学生考试的分数。我们先用上一小节学习的 if 语句来实现这个功能，在 LearnKotlin 文件中编写如下代码：

```
fun getScore(name: String) = if (name == "Tom") {
    86
} else if (name == "Jim") {
    77
} else if (name == "Jack") {
    95
} else if (name == "Lily") {
    100
} else {
    0
}
```

这里定义了一个 getScore()函数，这个函数接收一个学生姓名参数，然后通过 if 判断找到该学生对应的考试分数并返回。可以看到，这里再次使用了单行代码函数的语法糖，正如我所说，它真的很常用。

虽然上述代码确实可以实现我们想要的功能，但是写了这么多的 if 和 else，你有没有觉得代码很冗余？没错，当你的判断条件非常多的时候，就是应该考虑使用 when 语句的时候，现在我们将代码改成如下写法：

```
fun getScore(name: String) = when (name) {
    "Tom" -> 86
    "Jim" -> 77
    "Jack" -> 95
    "Lily" -> 100
    else -> 0
}
```

怎么样？有没有感觉代码瞬间清爽了很多？另外你可能已经发现了，when 语句和 if 语句一样，也是可以有返回值的，因此我们仍然可以使用单行代码函数的语法糖。

when 语句允许传入一个任意类型的参数，然后可以在 when 的结构体中定义一系列的条件，格式是：

```
匹配值 -> { 执行逻辑 }
```