第三个编程概念更隐晦一点，来自我们前面讨论流处理能力时说的“几乎免费的并行”。你需要放弃什么吗？可能需要对传给流方法的行为的写法稍作改变。你的行为必须能够同时对不同的输入安全地执行。一般情况下这就意味着，你写代码时不能访问共享的可变数据。这些函数有时被称为“纯函数”或“无副作用函数”或“无状态函数”。前面说的并行只有在假定你的代码的多个副本可以独立工作时才能进行。但如果要写入的是一个共享变量或对象，这就行不通了；如果两个进程需要同时修改这个共享变量怎么办？

Java 8的流实现并行比Java现有的线程API更容易，因此，尽管可以使用synchronized来打破“不能有共享的可变数据”这一规则，但这相当于是在和整个体系作对，因为它使所有围绕这一规则作出的优化都失去意义了。在多个处理器内核之间使用synchronized，其代价往往比你预期的要大得多，因为同步迫使代码按照顺序执行，而这与并行处理的宗旨相悖。

这两个要点：

* 没有共享的可变数据
* 将方法和函数即代码传递给其他方法的能力

是我们平常所说的函数式编程范式的基石。与此相反，在命令式编程范式中，你写的程序规则是一系列改变状态的指令。“不能有共享的可变数据”的要求意味着，一个方法是可以通过它将参数值转换为结果的方式完全描述的；换句话说，它的行为就像一个数学函数，没有可见的副作用。