JavaScript函数式编程

什么是函数式编程

是相对于面向过程；面向对象编程而言的。

按照我的理解，这是将一个整个任务执行的过程看做一个流水线，单个函数就是流水线上的一个结点。

流水线上，每经过一个结点，原材料就向着最终的成品更进一步

而在函数式编程中，函数处理的是数据。每经过一个函数的处理，数据就朝着我们想要的形式更进一步

函数式编程的特点

1. 函数是一等公民

函数能够存在于任何变量能够去到的地方

比如可以作为返回值，可以作为参数

2. 惰性执行

只在需要的时候执行，不会产生无意义的中间变量。

3. 无状态

对于一个函数，给定相同的输入，一定能够得到相同的输出，不依赖外部状态的变化

This 指针；IO 操作；全局变量

4. 数据不可变

所有数据都是不可变的，如果想要修改一个对象，需要创建一个新的对象用来修改，而不是修改已有的对象。

无副作用

一个数据经过 map 处理后，不应该修改原数据

纯函数

1. 便于测试和优化

实际项目开发中意义非常重大。

可以轻松断言函数的执行结果，可以保证函数的优化不会影响其他代码的执行，十分符合 TDD（测试驱动开发）的思想，代码更加健壮

2. 可缓存性

3. 自文档化

没有副作用，依赖明确（函数签名），易于理解和观察，函数还便于添加 jsdoc 注释，辅助开发

4. 更少 BUG

不存在指向不明的引用，不会修改参数。

很多共享状态是 BUG 产生的来源

函数执行流水线的构建

柯里化

是流水线上的加工站

将一个多元函数转换成依次调用的单元函数

函数的返回值只有一个，那么函数的参数也就只能有一个，这样才能完成流水线的组装：让每个加工站的输出刚好流向下个加工站的输入

部分函数应用 vs 柯里化

高级柯里化

柯里化的应用

函数组合

由加工站组合而成，就是流水线本身

函数组合实例；理解

函数组合的应用

注意：.add() 这样的调用是面向对象才有的，并不属于函数式

实际上是一种 pipe 的概念。和 Linux 中的命令调用思想一致

函数组合的好处

让代码更具语意，更有表现力，可读性更强

编程经验

柯里化中把要操作的数据放到最后

函数组合中函数要求单输入

函数组合的 Debug

多参考 Ramda

函数式编程缺陷

性能：对函数过渡包装，上下文切换过程中产生额外的性能开销

在 JS 这种非函数式语言中，函数式比起直接的语句指令慢

资源占用：不允许修改状态，需要创建新的对象，对内存、gc 会造成较大的压力

递归陷阱

- JavaScript函数式编程
  - 什么是函数式编程
  - 函数式编程的特点
    - 纯函数
      - 1. 便于测试和优化
      - 2. 可缓存性
      - 3. 自文档化
      - 4. 更少 BUG
    - 1. 函数是一等公民
    - 2. 惰性执行
    - 3. 无状态
    - 4. 数据不可变
      - 无副作用
  - 函数执行流水线的构建
    - 柯里化
      - 部分函数应用 vs 柯里化
      - 高级柯里化
      - 柯里化的应用
    - 函数组合
      - 函数组合实例；理解
      - 函数组合的应用
        - 注意：.add() 这样的调用是面向对象才有的，并不属于函数式
        - 实际上是一种 pipe 的概念。和 Linux 中的命令调用思想一致
      - 函数组合的好处
  - 编程经验
    - 柯里化中把要操作的数据放到最后
    - 函数组合中函数要求单输入
    - 函数组合的 Debug
    - 多参考 Ramda
  - 函数式编程缺陷
    - 性能：对函数过渡包装，上下文切换过程中产生额外的性能开销
    - 资源占用：不允许修改状态，需要创建新的对象，对内存、gc 会造成较大的压力
    - 递归陷阱