**语言**

语言是一种交流的工具，这约定了语言的“工具”本质，以及“交流”的功用。“工具”的选择只在于“功用”是否能达到，而不在于工具是什么。

在数千年之前，远古祭师手中的神杖就是他们与神交流的工具。祭师让世人相信他们敬畏的是神，而世人只需要相信那柄神杖。于是，假如祭师不小心丢掉了神杖，就可以堂而皇之地再做一根。甚至，他们可以随时将旧的换成更新或更旧的神杖，只要他们宣称这是一根更有利于通神的杖。对此，世人往往做出迷惑的表情，或者欢欣鼓舞的情状。今天，这种表情或情状一样地出现在大多数程序员的脸上，出现在他们听闻到新计算机语言被创生的时刻。  
神杖换了，祭师还是祭师，世人还是会把头叩得山响。祭师掌握了与神交流的方法（如果真如同他们自己说的那样的话），而世人只看见了神杖。

所以，泛义的工具是文明的基础，而确指的工具却是愚人的器物。

计算机语言有很多种分类方法，例如高级语言或者低级语言。其中一种分类方法，就是“静态语言”和“动态语言”——事物就是如此，如果用一对绝对反义的词来分类，就相当于概含了事物的全体。当然，按照中国人中庸平和的观点，以及保守人士对未知可能性的假设，我们还可以设定一种中间态：半动态语言。你当然也可以叫它半静态语言，这个随便你。

所以，我们现在是在讨论一种很泛义的计算机语言工具。至少在眼下，它（在分类概念中）概含了计算机语言的二分之一。当然，限于我自身的能力，我只能讨论一种确指的工具，例如JavaScript。但我希望你由此看到的是计算机编程方法的基础，而不是某种愚人的器物。JavaScript的生命力可能足够顽强，我假定它比C还顽强，甚至比你我的生命都顽强。但它只是愚人的器物，因此反过来说：它能不能长久地存在都并不重要，重要的是它能不能作为这“二分之一的泛义”来供我们讨论。

二十年来，开发新编程语言的一个流行的秘诀是：取C语言的计算模式，逐渐地往上加LISP模式的特性，例如运行时类型和无用单元收集。

然而这毕竟只是“创生一种新语言”的魔法。那么，到底有没有让我们在这浩如烟海的语言家族中，找到学习方法的魔法呢？

我的答案是：看清语言的本质，而不是试图学会一门语言。当然，这看起来非常概念化。甚至有人说我可能是从某本教材中抄来的，另外一些人又说我试图在这本书里宣讲类似于我那本《大道至简》里的老庄学说 。  
其实这很冤枉。我想表达的意思不过是：如果你想把一副牌理顺，最好的法子，是回到它的分类法上，要么从A到K整理，要么按四个花色整理 。毕竟，两种或更多种分类法作用于同一事物，只会使事物混淆而不是弄得更清楚。

因此，本书从语言特性出发，把动态与静态、函数式与非函数式的语言特性分列出来。先讲述每种特性，然后再讨论如何去使用（例如交叉）它们。

**特性**

无论哪种语言（或其他工具）都有其独特的特性，以及借鉴自其他语言的特性。有些语言通体没有“独特特性”，只是另外一种语言的副本，这更多的时候是为了“满足一些人使用语言的习惯”。还有一些语言则基本上全是独特的特性，这可能导致语言本身不实用，但却是其他语言的思想库。

我们已经讨论过这一切的来源。

对于JavaScript来说，除了动态语言的基本特性之外，它还有着与其创生时代背景密切相关的一些语言特性。直到昨天 ，JavaScript的创建者还在小心翼翼地增补着它的语言特性。JavaScript轻量的、简洁的、直指语言本实的特性集设计，使它成为解剖动态语言的有效工具。这个特性集包括：

 一套参考过程式语言惯例的语法；  
 一套以原型继承为基础的对象系统；  
 一套支持自动转换的弱类型系统；  
 动态语言与函数式语言的基本特性。

avaScript是一门很有趣的语言，每取得一点更深入的理解，我便以为自己已经彻悟。从最初的函数使用、面向对象、原型继承、函数对象到后来作用域链、functional、预解析、表达式解析机制、动态语言特性，每在此道路上前进一步，我都曾以为自己到达了顶峰。然而，事实上是，简单的语法也能承载深刻的思想。巧妙的设计让JavaScript成为一门兼有元类（meta-class）、类（class-based）、原型（prototype-based）三种面向对象特性和functional特性的动态语言。回忆起来，对这样的语言，说我已到达顶峰，是十分可笑的。