

# Go 语言

## 核心编程

李文塔 / 著

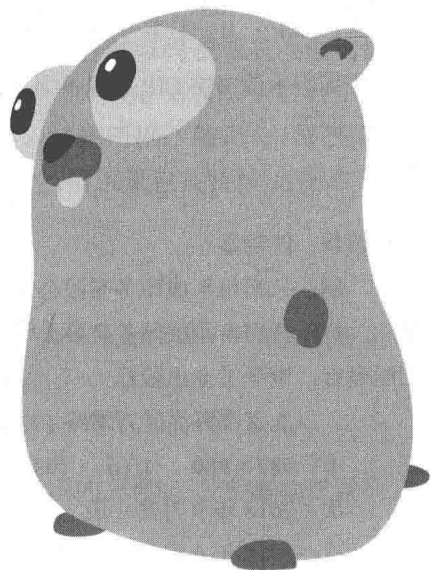


```
func main() {  
    c := make(chan string)  
    timeout := time.After(1 * time.Second)  
  
    go func() {  
        c <- "Hello, 世界"  
    }()  
  
    for {  
        select {  
        case result := <-c:  
            fmt.Println(result)  
  
        case <-timeout:  
            fmt.Println("Timeout")  
            return  
        }  
    }  
}
```

# Go 语言

## 核心编程

李文塔 / 著



電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

本书是一本系统介绍 Go 语言编程的书籍。首先介绍 Go 语言的基础知识，使读者对 Go 语言有一个整体的认知。接着围绕 Go 语言的三大语言特性：类型系统、接口和并发展开论述，本书不单单介绍每个语言特性怎么使用，在章节的最后还对重要语言特性的底层实现原理做了介绍。接着介绍反射这个高级语言特征。此外，本书专门用一章的篇幅来介绍 Go 语言的陷阱。最后介绍 Go 语言的工程实践和编程思想。相信本书能够帮助读者快速、深入地了解和学习这门语言。

本书适合各个层次的 Go 语言开发者阅读，初学者可以系统地从头学习，有一定的编程经验者可以选择性地阅读本书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

### 图书在版编目 (CIP) 数据

Go 语言核心编程 / 李文塔著. —北京：电子工业出版社，2018.9

ISBN 978-7-121-34911-9

I. ①G… II. ①李… III. ①程序语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字 (2018) 第 187936 号

责任编辑：陈晓猛

印 刷：三河市华成印务有限公司

装 订：三河市华成印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×980 1/16 印张：17.5 字数：336 千字

版 次：2018 年 9 月第 1 版

印 次：2018 年 9 月第 1 次印刷

定 价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 [zlt@phei.com.cn](mailto:zlt@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

本书咨询联系方式：010-51260888-819, [faq@phei.com.cn](mailto:faq@phei.com.cn)。

# 前言

## 写作背景

2007年，Go语言诞生于谷歌公司，2009年开源，2012年推出1.0版本，曾两次获得TIOBE年度语言（2009年和2016年）。2012年起，全球大量的开源项目开始使用Go语言进行开发，目前Go语言已经成为云计算领域事实上的标准语言，特别是在容器领域，诞生了一大批优秀的开源软件，如Docker、Kubernetes等。2017年区块链技术在国内大热，区块链两个大的技术平台以太坊（Ethereum）和超级账本（Hyperledger）与项目Fabric都是基于Go语言构建的。Go语言的应用领域逐步扩大，目前在区块链、云计算、中间件和服务器编程领域显现出明显的优势。Go语言最先在云计算公司盛行，随后大量的互联网初创企业将Go语言作为后台主要开发语言。目前，无论互联网公司里的独角兽，还是BAT（百度、阿里巴巴、腾讯），都已将Go语言作为其技术栈的重要组成部分。市场对于Go语言编程人才的需求量也在持续上升。

目前，Go语言编程类图书有如下几个类别。

- 基础全面型

七牛团队写作和翻译的几本书（《Go语言编程》《Go程序设计语言》等），系统地介绍了Go开发的基础知识，为Go语言在国内的推广做出了很大的贡献。

- 源码深入型

雨痕的《Go语言学习笔记》，上半部分是对Go语言基础知识的总结，下半部分对源码进行了分析，是学习Go语言内部原理非常好的参考书。

- 专业领域型

专注于介绍某个应用领域，比如谢孟军的《Go Web编程》，这类图书主要是介绍使用Go语言在某个领域进行开发的相关知识。

市面上Go语言的图书这么多了，为什么又要写一本，本书和市面上的图书有什么区别呢？这也是我写这本书之前就认真思考过的问题。我发现市面上缺少介绍Go语言核心特性和使用陷

阱的书。Go 语言核心特性包括类型系统、接口、并发，这三部分是 Go 语言最精华、最优美、最重要的语言特性。于是我就围绕这三个主题写了本书，书名就叫《Go 语言核心编程》。以类型系统为例，Go 语言中的类型涉及简单类型、复合类型、命名类型、未命名类型、静态类型、动态类型、底层类型、接口类型、具体类型及类型字面量等诸多概念，这些在先前的书籍里没有系统地介绍过，本书试图帮助读者梳理清楚这些类型的含义，建立概念，认清类型本质并学会使用；在并发编程中给出了 5 个并发编程范式并系统地介绍了 context 标准库，这也是本书特有的。

## 内容简介

本书共 9 章，围绕如下主题展开：基础知识（第 1 章）、语言核心（第 2 章~第 7 章）、工程管理（第 8 章）和编程哲学（第 9 章）。

## 基础知识

第 1 章介绍 Go 语言编程的基础知识。基础知识部分力求从系统性的角度讲述 Go 语言简洁的语法知识，试图帮助读者了解 Go 语言源程序基本结构，这是本书不同于其他书籍的地方。本章先对 Go 的源程序进行整体介绍，然后从词法分析的角度介绍 Go 语言的各个 token，接着介绍 Go 语言的基础语法：变量和常量、数据类型及控制结构，让读者学习 Go 语言的语法知识时可以“既见树木，又见森林”。

## 语言核心

本书的第 2~7 章讲解 Go 语言核心知识，这部分是本书最核心、最重要的部分，主要围绕如下 4 个主题展开。

- 类型系统

本书用 3 章的篇幅来讲解 Go 语言的类型系统，分别是第 2 章函数、第 3 章类型系统和第 4 章接口。类型系统是 Go 语言的灵魂，Go 语言小而美的类型系统是其区别于其他语言的显著特征。函数在 Go 语言中是“一等公民”，非侵入式的接口设计也很有特点。类型系统是本书的一条主要线索，几乎贯穿本书的所有章节，在反射章节里会对 Go 的类型系统做一个总结。掌握类型系统是写好 Go 程序的关键。

- 并发编程

第 5 章介绍 Go 并发编程，并发编程的原生支持是 Go 语言显著的语言特征。“不要通过共

享内存来通信，而是通过通信来共享内存”，本章首先讲解语言层面对并发的支持，接着讲解 context 标准库的知识，最后讲解常用的并发范式和 Go 并发模型。

- 反射

第 6 章介绍 Go 语言反射的相关知识。反射是 Go 语言的高级特性，反射是把“双刃剑”，用好了会有强大的威力，但由于其复杂性且易产生运行时 panic，因此反射又表现出脆弱性。Go 语言没有提供泛型支持，所以在开发框架和大型系统中反射是必须面对的一个技术点，市面上的图书对这部分的介绍不多，本书分类总结了 Go 的反射 API，尽可能梳理出条理清晰的 API 结构；同时以类型系统和接口底层实现为基础来介绍反射；最后介绍著名的 Web 框架 martini 中使用的依赖注入库 inject。

- 陷阱和习惯用法

第 7 章介绍 Go 语言陷阱和一些习惯用法，包括 Go 语言使用过程中容易出错和初学者容易困惑的知识点。Go 语言虽然设计简洁，但在某些方面还是有瑕疵的，比如 defer 陷阱、短变量的声明、range 临时变量共享等。同时 Go 又有其鲜明的书写格式和习惯用法，本章的最后会介绍这些内容。

## 工程管理

Go 语言是一门面向工程的语言，而不是学术界的产物，第 8 章介绍 Go 语言工程管理方面的知识，主要介绍 Go 的编程环境、命名空间和包管理的相关内容。

## 编程哲学

第 9 章讨论编程哲学，这部分内容不是介绍编程细节知识，读者可以把它当作散文来读。先对 Go 语言编程哲学做一个总结：少即是多，世界是并行的，组合优于继承，面向接口编程，等等，试图从哲学的角度来阐述这些设计思想的先进性。最后介绍 Go 语言的里程碑事件及未来的发展方向。

## 相关约定

本书使用 Go 代表 Go 语言的简写，使用 go 表示 Go 语言的并发关键字，使用 goroutine 代表 Go 语言中的并发执行体。通道和 chan 都代表 Go 语言的通信管道。切片和 slice 都表示 Go 语言内置的可变数组。本书使用实例或者类型实例表示具体类型的变量，而没有使用传统语言对象的概念。标准库和标准包会混用，都是指 Go 语言自带的标准库。复制和拷贝具有相同的

语义，都表示将数据复制一份并拷贝到另一处内存空间。本书是基于 Go 1.10.2 写作的，新版本的变动请读者以官方文档为准。截至本书出版时，Go 1.11 发布，官方推出了新的包管理工具 go mod，go mode 兼容 dep，dep 仍可放心使用，go mod 是新增的功能，可能需要一段时间才能成熟。

## 总结

本书绝大部分内容是我六年来在学习和实践 Go 语言过程中的总结和感悟，成书过程中也参阅了部分网上和官方资料，由于能力有限，疏忽和不足之处难免发生，欢迎读者指正，以便及时修订，我的邮箱是 [email.tata@qq.com](mailto:email.tata@qq.com)。

## 感谢

本书原稿基于 GitBook+Markdown 在 Linux 下使用 Vim 完成写作，原始稿件版式简单，非常感谢电子工业出版社博文视点编辑部的帮助，使其变成一本优美的纸质书。非常感谢曾刘刚通读全书，帮助审稿；同时感谢宫振飞、胡宸源、宋磊在成书过程中给予的宝贵意见。在时间碎片化的今天，一字一句地写完一本书是对自己毅力和耐心的极大考验，感谢在成书过程中妻子黄静给予的鼓励和支持。封面上可爱的地鼠来自上田拓也的作品，非常感谢上田先生欣然授权本书使用。

李文塔

### 读者服务

轻松注册成为博文视点社区用户 ([www.broadview.com.cn](http://www.broadview.com.cn))，扫码直达本书页面。

- **下载资源：**本书如提供示例代码及资源文件，均可在 [下载资源](#) 处下载。
- **提交勘误：**您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 [读者评论](#) 处留下您的疑问或观点，与其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/34911>



# 目录

|   |    |
|---|----|
| 第 1 章 基础知识.....                                   | 1  |
| 1.1 语言简介.....                                     | 1  |
| 1.1.1 Go 语言的诞生背景.....                             | 2  |
| 1.1.2 语言特性.....                                   | 2  |
| 1.1.3 Go 语言的特性.....                               | 4  |
| 1.1.4 总结.....                                     | 5  |
| 1.2 初识 Go 程序.....                                 | 5  |
| 1.3 Go 词法单元.....                                  | 6  |
| 1.3.1 token.....                                  | 7  |
| 1.3.2 标识符.....                                    | 8  |
| 1.3.3 操作符 ( operators ) 和分隔符 ( delimiters ) ..... | 12 |
| 1.3.4 字面常量.....                                   | 13 |
| 1.3.5 总结.....                                     | 15 |
| 1.4 变量和常量.....                                    | 16 |
| 1.4.1 变量.....                                     | 17 |
| 1.4.2 常量.....                                     | 18 |
| 1.5 基本数据类型.....                                   | 19 |
| 1.5.1 布尔类型.....                                   | 20 |
| 1.5.2 整型.....                                     | 21 |
| 1.5.3 浮点型.....                                    | 21 |
| 1.5.4 复数类型.....                                   | 21 |
| 1.5.5 字符串.....                                    | 22 |



|       |                      |    |
|-------|----------------------|----|
| 1.5.6 | rune 类型.....         | 23 |
| 1.6   | 复合数据类型 .....         | 23 |
| 1.6.1 | 指针.....              | 24 |
| 1.6.2 | 数组.....              | 25 |
| 1.6.3 | 切片.....              | 26 |
| 1.6.4 | map.....             | 29 |
| 1.6.5 | struct.....          | 31 |
| 1.7   | 控制结构 .....           | 32 |
| 1.7.1 | if 语句 .....          | 33 |
| 1.7.2 | switch 语句 .....      | 34 |
| 1.7.3 | for 语句 .....         | 36 |
| 1.7.4 | 标签和跳转.....           | 37 |
| <br>  |                      |    |
| 第 2 章 | 函数.....              | 40 |
| 2.1   | 基本概念 .....           | 41 |
| 2.1.1 | 函数定义.....            | 41 |
| 2.1.2 | 多值返回.....            | 42 |
| 2.1.3 | 实参到形参的传递.....        | 43 |
| 2.1.4 | 不定参数.....            | 43 |
| 2.2   | 函数签名和匿名函数 .....      | 45 |
| 2.2.1 | 函数签名 .....           | 45 |
| 2.2.2 | 匿名函数.....            | 47 |
| 2.3   | defer .....          | 48 |
| 2.4   | 闭包 .....             | 52 |
| 2.4.1 | 概念.....              | 52 |
| 2.4.2 | 闭包的价值.....           | 56 |
| 2.5   | panic 和 recover..... | 56 |
| 2.5.1 | 基本概念.....            | 56 |
| 2.5.2 | 使用场景.....            | 59 |
| 2.6   | 错误处理 .....           | 60 |
| 2.6.1 | error.....           | 60 |
| 2.6.2 | 错误和异常.....           | 61 |

|                                       |            |
|---------------------------------------|------------|
| 2.7 底层实现 .....                        | 63         |
| 2.7.1 函数调用规约 .....                    | 63         |
| 2.7.2 汇编基础 .....                      | 63         |
| 2.7.3 多值返回分析 .....                    | 64         |
| 2.7.4 闭包底层实现 .....                    | 68         |
| <b>第3章 类型系统 .....</b>                 | <b>72</b>  |
| 3.1 类型简介 .....                        | 73         |
| 3.1.1 命名类型和未命名类型 .....                | 73         |
| 3.1.2 底层类型 .....                      | 75         |
| 3.1.3 类型相同和类型赋值 .....                 | 75         |
| 3.1.4 类型强制转换 .....                    | 78         |
| 3.2 类型方法 .....                        | 80         |
| 3.2.1 自定义类型 .....                     | 80         |
| 3.2.2 方法 .....                        | 84         |
| 3.3 方法调用 .....                        | 87         |
| 3.3.1 一般调用 .....                      | 87         |
| 3.3.2 方法值 (method value) .....        | 88         |
| 3.3.3 方法表达式 (method expression) ..... | 89         |
| 3.3.4 方法集 (method set) .....          | 90         |
| 3.3.5 值调用和表达式调用的方法集 .....             | 92         |
| 3.4 组合和方法集 .....                      | 94         |
| 3.4.1 组合 .....                        | 94         |
| 3.4.2 组合的方法集 .....                    | 99         |
| 3.5 函数类型 .....                        | 101        |
| <b>第4章 接口 .....</b>                   | <b>105</b> |
| 4.1 基本概念 .....                        | 106        |
| 4.1.1 接口声明 .....                      | 106        |
| 4.1.2 接口初始化 .....                     | 107        |
| 4.1.3 接口方法调用 .....                    | 108        |
| 4.1.4 接口的动态类型和静态类型 .....              | 109        |

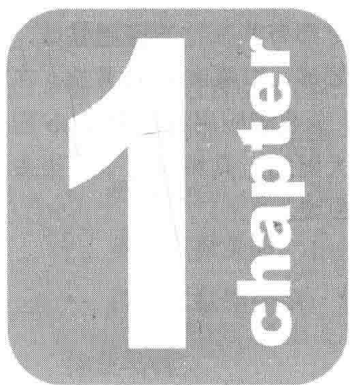
|              |                                 |            |
|--------------|---------------------------------|------------|
| 4.2          | 接口运算 .....                      | 110        |
| 4.2.1        | 类型断言 (Type Assertion) .....     | 110        |
| 4.2.2        | 类型查询 (Type Switches) .....      | 114        |
| 4.2.3        | 接口优点和使用形式 .....                 | 117        |
| 4.3          | 空接口 .....                       | 118        |
| 4.3.1        | 基本概念 .....                      | 118        |
| 4.3.2        | 空接口的用途 .....                    | 118        |
| 4.3.3        | 空接口和 nil .....                  | 118        |
| 4.4          | 接口内部实现 .....                    | 120        |
| 4.4.1        | 数据结构 .....                      | 120        |
| 4.4.2        | 接口调用过程分析 .....                  | 123        |
| 4.4.3        | 接口调用代价 .....                    | 128        |
| 4.4.4        | 空接口数据结构 .....                   | 131        |
| <b>第 5 章</b> | <b>并发 .....</b>                 | <b>133</b> |
| 5.1          | 并发基础 .....                      | 133        |
| 5.1.1        | 并发和并行 .....                     | 133        |
| 5.1.2        | goroutine .....                 | 134        |
| 5.1.3        | chan .....                      | 137        |
| 5.1.4        | WaitGroup .....                 | 140        |
| 5.1.5        | select .....                    | 142        |
| 5.1.6        | 扇入 (Fan in) 和扇出 (Fan out) ..... | 143        |
| 5.1.7        | 通知退出机制 .....                    | 143        |
| 5.2          | 并发范式 .....                      | 145        |
| 5.2.1        | 生成器 .....                       | 145        |
| 5.2.2        | 管道 .....                        | 150        |
| 5.2.3        | 每个请求一个 goroutine .....          | 151        |
| 5.2.4        | 固定 worker 工作池 .....             | 157        |
| 5.2.5        | future 模式 .....                 | 162        |
| 5.3          | context 标准库 .....               | 164        |
| 5.3.1        | context 的设计目的 .....             | 165        |
| 5.3.2        | 基本数据结构 .....                    | 165        |

|              |                    |            |
|--------------|--------------------|------------|
| 5.3.3        | API 函数             | 170        |
| 5.3.4        | 辅助函数               | 171        |
| 5.3.5        | context 的用法        | 173        |
| 5.3.6        | 使用 context 传递数据的争议 | 179        |
| 5.4          | 并发模型               | 179        |
| 5.4.1        | CSP 简介             | 179        |
| 5.4.2        | 调度模型               | 180        |
| 5.4.3        | 并发和调度              | 181        |
| <b>第 6 章</b> | <b>反射</b>          | <b>185</b> |
| 6.1          | 基本概念               | 186        |
| 6.1.1        | 基本数据结构和入口函数        | 186        |
| 6.1.2        | 基础类型               | 196        |
| 6.1.3        | 类型汇总               | 198        |
| 6.2          | 反射规则               | 200        |
| 6.2.1        | 反射 API             | 200        |
| 6.2.2        | 反射三定律              | 203        |
| 6.3          | inject 库           | 204        |
| 6.3.1        | inject 是什么         | 204        |
| 6.3.2        | 依赖注入和控制反转          | 204        |
| 6.3.3        | inject 实践          | 205        |
| 6.3.4        | inject 原理分析        | 207        |
| 6.4          | 反射的优缺点             | 211        |
| 6.4.1        | 反射的优点              | 211        |
| 6.4.2        | 反射的缺点              | 211        |
| 6.4.3        | 反射的最佳实践            | 211        |
| <b>第 7 章</b> | <b>语言陷阱</b>        | <b>212</b> |
| 7.1          | 多值赋值和短变量声明         | 212        |
| 7.1.1        | 多值赋值               | 212        |
| 7.1.2        | 短变量的声明和赋值          | 216        |
| 7.2          | range 复用临时变量       | 219        |

|              |                    |            |
|--------------|--------------------|------------|
| 7.3          | defer 陷阱 .....     | 222        |
| 7.4          | 切片困惑 .....         | 226        |
| 7.4.1        | 数组 .....           | 226        |
| 7.4.2        | 切片 .....           | 229        |
| 7.5          | 值、指针和引用 .....      | 234        |
| 7.5.1        | 传值还是传引用 .....      | 234        |
| 7.5.2        | 函数名的意义 .....       | 237        |
| 7.5.3        | 引用语义 .....         | 237        |
| 7.6          | 习惯用法 .....         | 238        |
| 7.6.1        | 干净与强迫症 .....       | 238        |
| 7.6.2        | comma,ok 表达式 ..... | 239        |
| 7.6.3        | 简写模式 .....         | 240        |
| 7.6.4        | 包中的函数或方法设计 .....   | 241        |
| 7.6.5        | 多值返回函数 .....       | 242        |
| <b>第 8 章</b> | <b>工程管理 .....</b>  | <b>243</b> |
| 8.1          | 编程环境 .....         | 243        |
| 8.1.1        | 环境搭建 .....         | 243        |
| 8.1.2        | 工程结构 .....         | 245        |
| 8.1.3        | 交叉编译 .....         | 248        |
| 8.2          | 命名空间和作用域 .....     | 250        |
| 8.2.1        | 命名空间 .....         | 250        |
| 8.2.2        | 作用域 .....          | 250        |
| 8.2.3        | 变量覆盖 .....         | 251        |
| 8.3          | 包的基本概念 .....       | 251        |
| 8.3.1        | 基本概念 .....         | 251        |
| 8.3.2        | 包引用 .....          | 252        |
| 8.3.3        | 包加载 .....          | 254        |
| 8.4          | 第三方包管理 .....       | 255        |
| 8.4.1        | vendor .....       | 255        |
| 8.4.2        | dep .....          | 256        |

---

|                     |     |
|---------------------|-----|
| 第9章 编程哲学.....       | 261 |
| 9.1 Go 语言设计哲学.....  | 261 |
| 9.1.1 少即是多.....     | 261 |
| 9.1.2 世界是并行的.....   | 262 |
| 9.1.3 组合优于继承.....   | 262 |
| 9.1.4 非侵入式的接口.....  | 262 |
| 9.1.5 总结.....       | 263 |
| 9.2 Go 语言发展史.....   | 264 |
| 9.2.1 站在巨人的肩膀上..... | 264 |
| 9.2.2 里程碑.....      | 264 |
| 9.3 未来.....         | 266 |
| 9.3.1 争议.....       | 266 |
| 9.3.2 Go 2.....     | 266 |



# 第 1 章

## 基础知识

本章介绍 Go 语言编程的基础知识。常听一些朋友抱怨：“看了某一门语言的编程书籍一百多页，还不清楚一个简单的程序是什么模样，看着后面的内容忘记了前面的内容”。本章不会罗列一个个知识点来介绍语法知识，而是从程序员视角介绍 Go 的基础语法知识。先介绍 Go 源代码的整体构成和特点，然后从编译器词法产生 tokens 的视角来分析 Go 源程序的构成，最后介绍语法的基本知识。相信这种从整体到详细的论述视角能更好地帮助读者学习 Go 语言基础知识。

1.1 节先对 Go 语言的语言特性做一个整体的概述，使读者对 Go 语言有一个基本的印象。1.2 节通过一个简单的“hello,world”程序介绍 Go 语言源程序的整体构成。1.3 节从编译器编译源代码的角度介绍 Go 源程序中 token 的概念，并对每种 token 做一个简单的介绍，让读者对 Go 语言源程序的基本构成单元 token 有一个整体认识，通过 1.2 节和 1.3 节的介绍能让读者先对 Go 程序建立整体印象。1.4 节开始对 Go 语言源程序重要构成元素变量和常量进行介绍；接着介绍 Go 语言的基本数据类型和复合数据类型，最后介绍 Go 语言的控制结构。通过本章的学习，读者能够对 Go 语言的基础语法支持有一个整体的了解，为深入学习其他语言特性奠定基础。

### 1.1 语言简介

已经有那么多编程语言了，为什么还要发明新语言？为什么还要去学习新语言？相信不

少人都有这样的疑问。答案很简单，虽然有那么多语言，但每种语言都有其独特的应用领域，在某个领域使用某种语言能达到收益/投入的最大化。比如在嵌入式领域，汇编和 C 是首选；在操作系统领域，C 是首选；在系统级服务编程领域，C++ 是首选；在企业级应用程序和 Web 应用领域，Java 是首选。就好比木工的工具箱中锤子可以有很多种，大厨的工具箱中刀子有很多种一样，某种语言就像某种锤子或者某种刀一样，有其特别应用的领域。

### 1.1.1 Go 语言的诞生背景

Go 语言的诞生主要基于如下原因：

(1) 摩尔定律接近失效后多核服务器已经成为主流，当前的编程语言对并发的支持不是很好，不能很好地发挥多核 CPU 的威力。

(2) 程序规模越来越大，编译速度越来越慢，如何快速地编译程序是程序员的迫切需求。

(3) 现有的编程语言设计越来越复杂，由于历史的包袱，某些特性的实现不怎么优雅，程序员花费了更多的精力来应对编程语法细节而不是问题域。

Go 语言就是为了解决当下编程语言对并发支持不友好、编译速度慢、编程复杂这三个问题而诞生的。

### 1.1.2 语言特性

编程语言有几百种，语法形式千差万别，将这些语法进行抽象概括，剔除表现形式的差异，就形成了一个表达语义的语言特性，有些语言特性是某个语言独有的，但绝大部分语言特征是很多语言共有的。总的语言特性就几十种，下面归纳一下常用的高级语言的语言特性。

#### 语言组织

所有的高级语言都使用源代码来表达程序，程序的语法格式千差万别，但表达这些语法的基本概念大同小异，主要包括：

- 标识符和关键字；
- 变量和常量；
- 运算符；
- 表达式；
- 简单语句；
- 控制结构。



## 类型系统

每种高级语言都有自己的类型系统，类型系统的特性主要表现在以下几个方面。

- 动静特性：动态语言还是静态语言。
- 类型强弱：强类型还是弱类型。
- 基本数据类型：包括类型及其支持的运算和操作集合。
- 自定义数据类型：包括类型及其支持的运算和操作集合。

## 抽象特性

- 函数：是否支持函数、匿名函数、高阶函数、闭包等。
- 面向对象：是否支持面向对象。
- 多态：如何支持多态。
- 接口：是否支持接口，以及接口实现模式。

## 元编程特性

- 泛型：是否支持泛型。
- 反射：是否支持反射，反射的能力。

## 运行和跨平台语言特性

- 编译模式：是编译成可执行程序，还是编译成中间代码，还是解释器解释执行。
- 运行模式：直接由 OS 加载运行，还是由虚拟机加载执行。
- 内存管理：是否支持垃圾回收。
- 并发支持：是否原生支持并发，还是库支持。
- 交叉编译：是否支持交叉编译。
- 跨平台支持：是否支持多个平台。

## 语言软实力特性

- 库：标准库和第三方库是否丰富、好用、高效。
- 框架：是否有非常出众的框架。
- 语言自身兼容性：语言规范是否经常变换，语言新版本向前兼容性。
- 语言影响力：是否有商业公司支持，社区的活跃性，是否是著名项目。