在 XO 公司，我们最初使用 Node 和 Ruby 构建相互连接的服务系统。我们享受 Node 带来的明显性能优势，以及可以访问已有的大型软件包仓库。我们也可以轻松地在公司内部发布并复用已有的插件和模块。极大地提高了开发效率，使得我们可以快速编写出可拓展的和可靠的应用。而且，庞大的 Node 社区使我们的工程师向开源软件贡献更加容易（比如 [BunnyBus](https://link.zhihu.com/?target=https%3A//medium.com/xo-tech/bunnybus-building-a-data-transit-system-b9647f6283e5%23.l64fdvfys) 和 [Felicity](https://link.zhihu.com/?target=https%3A//medium.com/xo-tech/introducing-felicity-7b6d0b734ce%23.hmloiiyx8)）。

虽然我在大学时期和刚刚工作的一些时间在使用更严谨的编译语言，比如 C++ 和 C#，而后来我开始使用 JavaScript。我很喜欢它的自由和灵活，但是我最近开始怀念静态和结构化的语言，因为当时有一个同事让我对 Go 语言产生了兴趣。

我从写 JavaScript 到写 Go，我发现两种语言有很多相似之处。两者学习起来都很快并且易于上手，都具有充满表现力的语法，并且在开发者社区中都有很多工作机会。没有完美的编程语言，所以你应该总是选择一个适合手头项目的语言。在这篇文章中，我将要说明这两种语言深层次上的关键区别，希望能鼓励没有用过 Go 语言的用户可以有机会使用 Go 。

**大体上的差异**

在深入细节之前，我们应该先了解一下两种语言之间的重要区别。

Go，或称 Golang，是 Google 在 2007 年创建的自由开源编程语言。它以快速和简单为设计目标。Go 被直接编译成机器码，这就是它速度的来源。使用编译语言调试是相当容易的，因为你可以在早期捕获大量错误。Go 也是一种强类型的语言，它有助于数据完整，并可以在编译时查找类型错误。

另一方面，JavaScript 是一种弱类型语言。除了忽略验证数据的类型和真值判断陷阱所带来的额外负担之外，使用弱类型语言也有自己的好处。比起使用接口*interfaces*和范型*generics*，柯里化*currying*和可变的形参个数*flexible arity*让函数变得更加灵活。JavaScript 在运行时进行解释，这可能导致错误处理和调试的问题。Node 是一款基于 Google V8 虚拟机的 JavaScript 运行库，这使它成为一个轻量和快速的 Web 开发平台。

**语法**

作为原来的 JavaScript 开发者，Go 简单和直观的语法很吸引我。由于两种语言的语法可以说都是从 C 语言演变而来的，所以它们的语法有很多相同之处。Go 被普遍认为是一种“容易学习的语言”。那是因为它的对开发者友好的工具、精简的语法和固守惯例（LCTT 译注：惯例优先）。

Go 包含大量有助于简化开发的内置特性。你可以用标准 Go 构建工具把你的程序用 go build 命令编译成二进制可执行文件。使用内置的测试套件进行测试只需要运行 go test 即可。 诸如原生支持的并发等特性甚至在语言层面上提供。

[Google 的 Go 开发者](https://link.zhihu.com/?target=https%3A//golang.org/doc/faq)认为，现在的编程太复杂了，太多的“记账一样，重复劳动和文书工作”。这就是为什么 Go 的语法被设计得如此简单和干净，以减少混乱、提高效率和增强可读性。它还鼓励开发人员编写明确的、易于理解的代码。Go 只有 [25 个保留关键字](https://link.zhihu.com/?target=https%3A//golang.org/ref/spec%23Keywords)和一种循环（for 循环），而不像 JavaScript 有 [大约 84 个关键字](https://link.zhihu.com/?target=https%3A//www.w3schools.com/js/js_reserved.asp)（包括保留关键字字、对象、属性和方法）。

为了说明语法的一些差异和相似之处，我们来看几个例子：

* 标点符号： Go 去除了所有多余的符号以提高效率和可读性。尽管 JavaScript 中需要符号的地方也不多（参见： [Lisp](https://link.zhihu.com/?target=https%3A//en.wikipedia.org/wiki/Lisp_%2528programming_language%2529)），而且经常是可选的，但我更加喜欢 Go 的简单。

// JavaScript 的逗号和分号

for (var i = 0; i < 10; i++) {

console.log(i);

}

//JavaScript 中的标点

// Go 使用最少数量标点

for i := 0; i < 10; i++ {

fmt.Println(i)

}

// Go 中的标点

* 赋值：由于 Go 是强类型语言，所以你在初始化变量时可以使用 := 操作符来进行类型推断，以避免[重复声明](https://link.zhihu.com/?target=https%3A//golang.org/doc/faq%23principles)，而 JavaScript 则在运行时声明类型。

// JavaScript 赋值

var foo = "bar";

// JavaScript 中的赋值

// Go 的赋值

var foo string //不使用类型推导

foo = "bar"

foo := "bar" //使用类型推导

// Go 的赋值

* 导出：在 JavaScript 中，你必须从某个模块中显式地导出。 在 Go 中，任何大写的函数将被默认导出。

const Bar = () => {};

module.exports = {

Bar

}

// JavaScript 中的导出

// Go 中的导出

package foo // 定义包名

func Bar (s string) string {

// Bar 将被导出

}

// Go 中的导出

* 导入：在 JavaScript 中 required 库是导入依赖项和模块所必需的，而 Go 则利用原生的 import 关键字通过包的路径导入模块。另一个区别是，与 Node 的中央 NPM 存储库不同，Go 使用 URL 作为路径来导入非标准库的包，这是为了从包的源码仓库直接克隆依赖。

// Javascript 的导入

var foo = require('foo');

foo.bar();

// JavaScript 的导入

// Go 的导入

import (

"fmt" // Go 的标准库部分

"github.com/foo/foo" // 直接从仓库导入

)

foo.Bar()

// Go 的导入

* 返回值：通过 Go 的多值返回特性可以优雅地传递和处理返回值和错误，并且通过传递引用减少了不正确的值传递。在 JavaScript 中需要通过一个对象或者数组来返回多个值。

// Javascript - 返回多值

function foo() {

return {a: 1, b: 2};

}

const { a, b } = foo();

// JavaScript 的返回

// Go - 返回多值

func foo() (int, int) {

return 1, 2

}

a, b := foo()

// Go 的返回

* 错误处理：Go 推荐在错误出现的地方捕获它们，而不是像 Node 一样在回调中让错误冒泡。

// Node 的错误处理

foo('bar', function(err, data) {

// 处理错误

}

// JavaScript 的错误处理

//Go 的错误处理

foo, err := bar()

if err != nil {

// 用 defer、 panic、 recover 或 log.fatal 等等处理错误.

}

// Go 的错误处理

* 可变参数函数：Go 和 JavaScript 的函数都支持传入不定数量的参数。

function foo (...args) {

console.log(args.length);

}

foo(); // 0

foo(1, 2, 3); // 3

// JavaScript 中的可变参数函数

func foo (args ...int) {

fmt.Println(len(args))

}

func main() {

foo() // 0

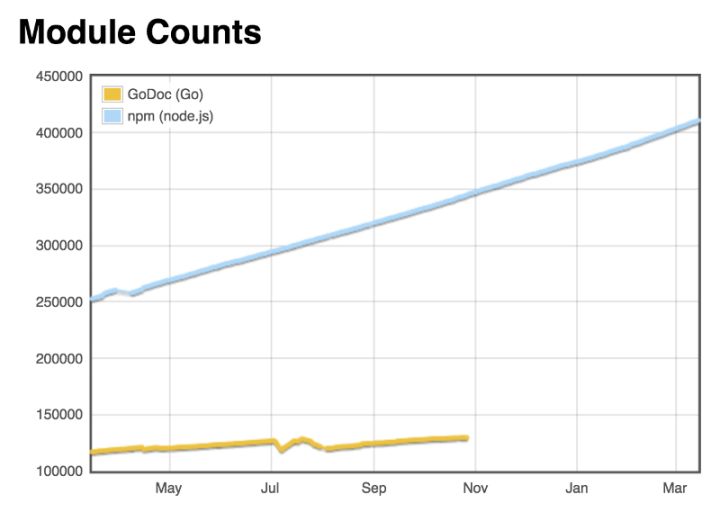
foo(1,2,3) // 3

}

// Go 中的可变参数函数

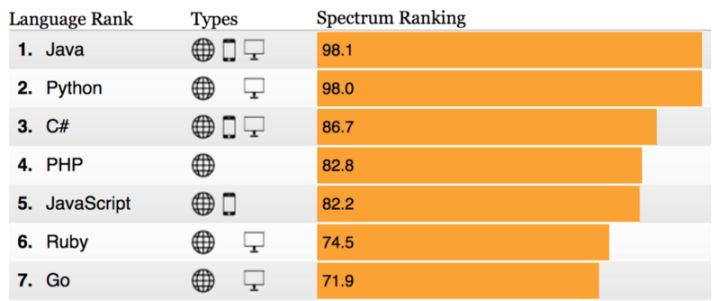
**社区**

当比较 Go 和 Node 提供的编程范式哪种更方便时，两边都有不同的拥护者。Node 在软件包数量和社区的大小上完全胜过了 Go。Node 包管理器（NPM），是世界上最大的软件仓库，拥有[超过 410,000 个软件包，每天以 555 个新软件包的惊人速度增长](https://link.zhihu.com/?target=http%3A//www.modulecounts.com/)。这个数字可能看起来令人吃惊（确实是），但是需要注意的是，这些包许多是重复的，且质量不足以用在生产环境。 相比之下，Go 大约有 13 万个包。



*Node 和 Go 包的数量*

尽管 Node 和 Go 岁数相仿，但 JavaScript 使用更加广泛，并拥有巨大的开发者和开源社区。因为 Node 是为所有人开发的，并在开始的时候就带有一个强壮的包管理器，而 Go 是特地为 Google 开发的。下面的[Spectrum 排行榜](https://link.zhihu.com/?target=http%3A//spectrum.ieee.org/static/interactive-the-top-programming-languages-2016)显示了当前流行的的顶尖 Web 开发语言。



*Web 开发语言排行榜前 7 名*

JavaScript 的受欢迎程度近年来似乎保持相对稳定，而 [Go 一直在保持上升趋势](https://link.zhihu.com/?target=http%3A//www.tiobe.com/tiobe-index/)。



*编程语言趋势*

**性能**

如果你的主要关注点是速度呢？当今似乎人们比以前更重视性能的优化。用户不喜欢等待信息。 事实上，如果网页的加载时间超过 3 秒，[40％ 的用户会放弃访问您的网站](https://link.zhihu.com/?target=https%3A//hostingfacts.com/internet-facts-stats-2016/)。

因为它的非阻塞异步 I/O，Node 经常被认为是高性能的语言。另外，正如我之前提到的，Node 运行在针对动态语言进行了优化的 Google V8 引擎上。而 Go 的设计也考虑到速度。[Google 的开发者们](https://link.zhihu.com/?target=https%3A//golang.org/doc/faq)通过建立了一个“充满表现力而轻量级的类型系统；并发和垃圾回收机制；强制地指定依赖版本等等”，达成了这一目标。

我运行了一些测试来比较 Node 和 Go 之间的性能。这些测试注重于语言提供的初级能力。如果我准备测试例如 HTTP 请求或者 CPU 密集型运算，我会使用 Go 语言级别的并发工具（goroutines/channels）。但是我更注重于各个语言提供的基本特性（参见 [三种并发方法](https://link.zhihu.com/?target=https%3A//medium.com/xo-tech/concurrency-in-three-flavors-51ed709876fb%23.khvqrttxa) 了解关于 goroutines 和 channels 的更多知识）。

我在基准测试中也加入了 Python，所以无论如何我们对 Node 和 Go 的结果都很满意。

**循环/算术**

迭代十亿项并把它们相加：

var r = 0;

for (var c = 0; c < 1000000000; c++) {

r += c;

}

// Node

package main

func main() {

var r int

for c := 0; c < 1000000000; c++ {

r += c

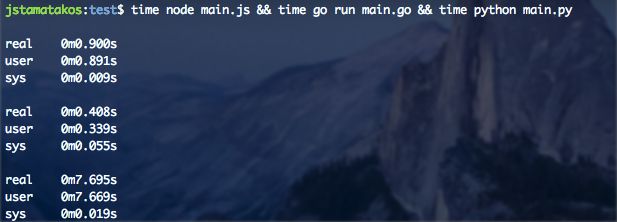
}

}

// Go

sum(xrange(1000000000))

// Python



*结果*

这里的输家无疑是 Python，花了超过 7 秒的 CPU 时间。而 Node 和 Go 都相当高效，分别用了 900 ms 和 408 ms。

*修正：由于一些评论表明 Python 的性能还可以提高。我更新了结果来反映这些变化。同时，使用 PyPy 大大地提高了性能。当使用 Python 3.6.1 和 PyPy 3.5.7 运行时，性能提升到 1.234 秒，但仍然不及 Go 和 Node 。*

**I/O**

遍历一百万个数字并将其写入一个文件。

var fs = require('fs');

var wstream = fs.createWriteStream('node');

for (var c = 0; c < 1000000; ++c) {

wstream.write(c.toString());

}

wstream.end();

// Node

package main

import (

"bufio"

"os"

"strconv"

)

func main() {

file, \_ := os.Create("go")

b := bufio.NewWriter(file)

for c := 0; c < 1000000; c++ {

num := strconv.Itoa(c)

b.WriteString(num)

}

file.Close()

}

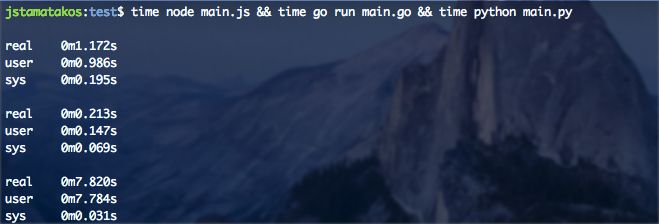
// Go

with open("python", "a") as text\_file:

for i in range(1000000):

text\_file.write(str(i))

// Python



*结果*

Python 以 7.82 秒再次排名第三。 这次测试中，Node 和 Go 之间的差距很大，Node 花费大约 1.172 秒，Go 花费了 213 毫秒。真正令人印象深刻的是，Go 大部分的处理时间花费在编译上。如果我们将代码编译，以二进制运行，这个 I/O 测试仅花费 78 毫秒——要比 Node 快 15 倍。

*修正：修改了 Go 代码以实现缓存 I/O。*

**冒泡排序**

将含有十个元素的数组排序一千万次。

function bubbleSort(input) {

var n = input.length;

var swapped = true;

while (swapped) {

swapped = false;

for (var i = 0; i < n; i++) {

if (input[i - 1] > input [i]) {

[input[i], input[i - 1]] = [input[i - 1], input[i]];

swapped = true;

}

}

}

}

for (var c = 0; c < 1000000; c++) {

const toBeSorted = [1, 3, 2, 4, 8, 6, 7, 2, 3, 0];

bubbleSort(toBeSorted);

}

// Node

package main

var toBeSorted [10]int = [10]int{1, 3, 2, 4, 8, 6, 7, 2, 3, 0}

func bubbleSort(input [10]int) {

n := len(input)

swapped := true

for swapped {

swapped = false

for i := 1; i < n; i++ {

if input[i-1] > input[i] {

input[i], input[i-1] = input[i-1], input[i]

swapped = true

}

}

}

}

func main() {

for c := 0; c < 1000000; c++ {

bubbleSort(toBeSorted)

}

}

// Go

def bubbleSort(input):

length = len(input)

swapped = True

while swapped:

swapped = False

for i in range(1,length):

if input[i - 1] > input[i]:

input[i], input[i - 1] = input[i - 1], input[i]

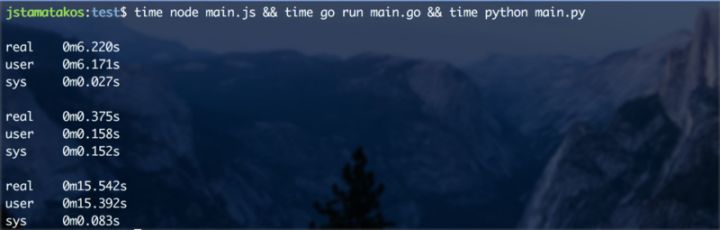
swapped = True

for i in range(1000000):

toBeSorted = [1, 3, 2, 4, 8, 6, 7, 2, 3, 0]

bubbleSort(toBeSorted)

//Python



*结果*

像刚才一样，Python 的表现是最差的，大约花费 15 秒完成了任务。 Go 完成任务的速度是 Node 的 16 倍。

**判决**

Go 无疑是这三个测试中的赢家，而 Node 大部分表现都很出色。Python 也表现不错。要清楚，性能不是选择编程语言需要考虑的全部内容。如果您的应用不需要处理大量数据，那么 Node 和 Go 之间的性能差异可能是微不足道的。 有关性能的一些其他比较，请参阅以下内容：

* [Node Vs. Go](https://link.zhihu.com/?target=https%3A//jaxbot.me/articles/node-vs-go-2014)
* [Multiple Language Performance Test](https://link.zhihu.com/?target=https%3A//hashnode.com/post/comparison-nodejs-php-c-go-python-and-ruby-cio352ydg000ym253frmfnt70)
* [Benchmarks Game](https://link.zhihu.com/?target=https%3A//benchmarksgame.alioth.debian.org/u64q/compare.php%3Flang%3Dgo%26amp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Bamp%3Blang2%3Dnode)

**结论**

这个帖子不是为了证明一种语言比另一种语言更好。由于各种原因，每种编程语言都在软件开发社区中占有一席之地。 我的意图是强调 Go 和 Node 之间的差异，并且促进展示一种新的 Web 开发语言。 在为一个项目选择语言时，有各种因素需要考虑，比如开发人员的熟悉程度、花费和实用性。 我鼓励在决定哪种语言适合您时进行一次彻底的底层分析。

正如我们所看到的，Go 有如下的优点：接近底层语言的性能，简单的语法和相对简单的学习曲线使它成为构建可拓展和安全的 Web 应用的理想选择。随着 Go 的使用率和社区活动的快速增长，它将会成为现代网络开发中的重要角色。话虽如此，我相信如果 Node 被正确地实现，它正在向正确的方向努力，仍然是一种强大而有用的语言。它具有大量的追随者和活跃的社区，使其成为一个简单的平台，可以让 Web 应用在任何时候启动和运行。

**资料**

如果你对学习 Go 语言感兴趣，可以参阅下面的资源：

* [Golang 网站](https://link.zhihu.com/?target=https%3A//golang.org/doc/%23learning)
* [Golang Wiki](https://link.zhihu.com/?target=https%3A//github.com/golang/go/wiki/Learn)
* [Golang Subreddit](https://link.zhihu.com/?target=https%3A//www.reddit.com/r/golang/)

via: [https://medium.com/xo-tech/from-node-to-go-a-high-level-comparison-56c8b717324a#.byltlz535](https://link.zhihu.com/?target=https%3A//medium.com/xo-tech/from-node-to-go-a-high-level-comparison-56c8b717324a%23.byltlz535)

作者：[John Stamatakos](https://link.zhihu.com/?target=https%3A//medium.com/%40johnstamatakos%3Fsource%3Dpost_header_lockup) 译者：[trnhoe](https://link.zhihu.com/?target=https%3A//github.com/trnhoe) 校对：[wxy](https://link.zhihu.com/?target=https%3A//github.com/wxy)

本文由 [LCTT](https://link.zhihu.com/?target=https%3A//github.com/LCTT/TranslateProject) 原创编译，[Linux中国](https://link.zhihu.com/?target=https%3A//linux.cn/) 荣誉推出