如果你做过Web开发，你可能会觉得我们正处于一个新时代的开端。多年以来我们一直使用像Django Rails 这些基于Python、Ruby、PHP开发的框架，这些开发语言都是基于老式的“请求-应答周期”模式（request-response cycle），之所以用这个短语是因为找不到更好的专业术语。

这个模式主要是这样的：一个请求从客户端发送到服务端，然后服务端渲染一个HTML页面，接着将这个页面返回给客户端。虽然随着AJAX、JSON和大量内置了客户端渲染功能的框架（例如Ember、Angular、Knockout、Backone和最近的React）的出现，这个模式不像10年前那样在web开发里占有统治地位了，但是在很多情况下我们依然使用一些不适合这个模式的开发语言和框架。

很多年前，Python和Ruby都比较慢，但是我们都乐意使用他们。但是随着我们越来越将服务端设计为可以将JSON发送给不同客户端的REST API，对大多数人来说，是时候为它们（指Python、Ruby）找一个替代品了。

你看见很多竞争者试图填补这个空隙。一些异想天开的开发者使用Rust、Nim或者Haskell来完成这个工作，而另外一些开发者则使用基于JVM的语言，例如Scala或者Clojure（因为JVM把多线程问题处理得很好），但是目前为止讨论得最多，也是黑得最严重的是NodeJS和Go。它俩都不优雅。

在ES6之前，JavaScript的很多地方就是一坨翔，但是目前很多对JavaScirpt的偏见都是基于之前的认识而不是现在的。正如我之前所说，我个人认为JavaScript完全可以接受，特别是当你使用一些基于它的超集，例如Typescript的时候。JavaScript的优点是它相对较快，并且已经在浏览器中使用（这个可能会在将来使用 WebAssembly 之后有所改变），以及天生就是异步的，这非常适合用来做现代的web开发（web app的开发不仅仅限于HTML页面）。

虽然JavaScript有一些缺陷，特别是语法非常丑陋繁琐。但它持续不断地开展标准化进程，这又让它变得越来越好，也是我最欣赏它的地方。JavaScript的一些改进是必须的，这让它变得可用。随着许多机构都插手ES的标准化过程，有些人担忧JavaScript会变得特别地臃肿。这也是我所担忧的。事情最好是这样，有个单一的力量在背后推动JavaScript的发展，但是我不确定JavaScript是否会变成这样。或者说如果变成这样了，这个力量是否有能力去维护JavaScript。Microsoft、Google、Mozilla以及其它机构都参与进来其实是一把双刃剑。

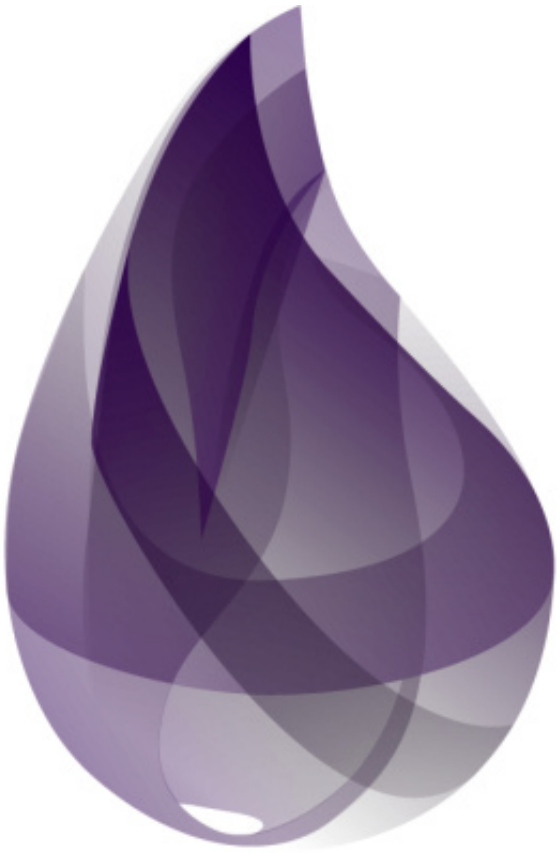
发展到最后，JavaScript能够非常好地处理现代的web开发，但它也不是完美的。如果一个请求非常消耗CPU，那么系统中的每个用户都需要等待。虽然Promise对象减少了回调地狱的出现，但是从根本上来说，我们依然在掩饰JavaScript核心语法的缺陷并视图修复它。JavaScript是否会很好地处理多线程问题从而使上述问题得到解决？Web Workers是IPC的一种形式，如果你用这个模型来处理并发，那么JavaScript并不比Python走得更远（我知道aysncio，但我这里是说GIL）。

目前只有少数语言真正适合用来处理并发，其中之一就是Go，它使用goroutines和channels，这是它主要吸引人的地方。Go的其它有点还有快速、简单、动态类型、编译后的文件是简单的、最终的（译注：这里作者是说Go语言编译后的文件简单，没有繁琐的依赖，除了C语言的一些基本库外不依赖其它库，可以立即独立执行）。

在关于Go的讨论中，这些优点似乎成了一条经常被忽视的分界线。很多使用过、或者还在使用更加现代的语言（例如Haskll）的开发者觉得Go很恐怖，因为它缺乏一些特性。并且无法理解你为什么要用Go自缚手脚。但在另一方面，如果你真的听了一些Go用户的言论，你会知道这就是Go卖点：语言必须精简，从而可以在短时间内了解。并且无论它缺少了什么，你可以写大量的的代码暴力搞定。这很烦但有效。我用过一些更加小众难懂语言做开发，因此我不是Go的脑残粉，很多Go的使用者其实都在无聊的语言争论中扮演了脑残粉的角色。但是在十数年的编程生涯中用过多种语言后，我只是期待一种简单并且适合自己的语言。这是一种我愿意的让步，并且我肯定很多Go的用户也有相同的想法。

正如所说的那样，你试过用Go开发web应用程序吗？你可以这么做，但是过程并不愉快。Python和Ruby中那些优秀的处理表单的库？没错，它们都不怎么好。你可以试着写一些函数为不同的表单输入做验证，但是你可能会受到类型系统的限制，并且会发现有有些事情不能像你之前用的语言那样容易做。数据库处理则更加繁琐，带有标签的模型对于JSON和数据库等等来说非常丑陋。Go不是一个理想的解决方案。我喜欢简洁的事物，但是web开发非常繁琐，而Go只会令事情恶化。

在一些关于语言的讨论中，Go经常被黑，因为它一开始就被标榜为“系统语言”，然而说Go会取代C或者C++是荒谬的。Go的开发者曾经非常惊讶，因为许多Go的改进是来自动态语言例如Python而不是C或者C++。当“系统语言”这个词用来区分不同的开发语言的时候，它并没有什么意义。但是在Google内部，在有着大量服务器和系统的环境中，“系统语言”这个词就有意义了。 老实说，Go确实适合用来开发一些应用程序、基础程序（如Docker）和小的命令行工具。但是在其它地方，例如开发web应用程序，我不认为Go是合适的。因为这超出了它所谓的“系统语言”的范围。



（图：Elixir 语言的 Logo）

然而有一门语言符合上述的要求，即使是挑剔的人和追求简单的人都会为它而兴奋。假设你看过这篇文章的标题，那么你知道我说的就是Elixir （译注：Elixir 的音标是 /ɪ’lɪksɚ/ ）。Elixir是一门非常轻量级的类Ruby语言，它最终会编译到BEAM平台上运行，即 Erlang虚拟机。Erlang是爱立信公司创建的，在Erlang上面运行的系统的持续运行时间达到了难以置信的几十年。你可以去搜索一下为何它如此厉害，其中一个因数是Erlang的进程是非常轻量级的，成百上千个进程可以在一瞬间就被创建出来。这是并发的真谛，也相当优雅。它允许像WhatsApp这样的应用以极少的服务器和工程师下运行。

我不想说太多关于并发的事，主要是因为其他人会做得更好。你真正需要知道的是，在并发的问题上，Erlang/Elixir 比其它所有语言都做得好。我主要想说的是Elixir它自身的质量。

你想要一个现代的编程语言吗？Elixir是函数式的、不可变的并且支持模式匹配，这就像给代码打上了激素（译注：形容Elixir写起来很爽）。仅仅这些解释并不会对你的编程观有什么影响。Elixir还支持宏，这意味着语言的核心可以保持很小，但是用户可以扩展它以支持一些即

使是语言设计者也没有想到的模式。正如我之前所说，Elixir使用了一套类Ruby的语法实现了这些东西。语法不应该是问题的关键，不过此时它确实是。我之前试图在语法中夹杂一些诗词（我觉得这非常酷）来设计一个可以让人接受的list语言（或者说是设计一套可以让自己接受的语法），但是Elixir简洁的核心语法和天生就是函数式编程让我震惊了，它简直就是Scheme的继承人。

这也是Elixir最吸引我的地方，尽管它是一个非常现代的语言，但是它非常小。你可以在几小时内就通读一次语言指南并且对核心概念有一个很好的理解。我喜欢这样。我更加喜欢花时间去解决问题而不是花时间去指出在这门语言中应该使用什么概念。这也能花费更少的时间去掌握语言。将10个概念每个使用100次，将100个概念每个使用10次，你会直观地发现前者明显比后者快。

因此我的意思是？我认为我们将处于一个多语言编程的时代。如果你之前打算多学几门开发语言，这会变得越来越来不现实。使用Elixir来实现REST API服务器，你很可能依然需要调用Python（或者其它语言）脚本来完成其他任务。但是重点是，Elixir的框架（例如Phoneix）和类库（例如Rcto）似乎比其它可选的语言更加适合web开发。如果Elixir唯一的能做好的事情（事实上它确实做好了）是连接到Postgres数据库，然后将数据转化为JSON，那么这种模式符合Unix的哲学（程序只做一件事，并且把它做好）。

如果我们处于这样的一个境地：用户为了找到更加适合的工具，舍得放弃他们所用的编程语言中大量的类库。那么Elixir似乎是一个比Go更加合适的选择。两者都是简洁的语言，但是Elixir有大量的特性，这不仅让你像一个程序员那样成长，并且让你的程序随着时间变得优雅。