Go

Why Golang?

(this is a great article)

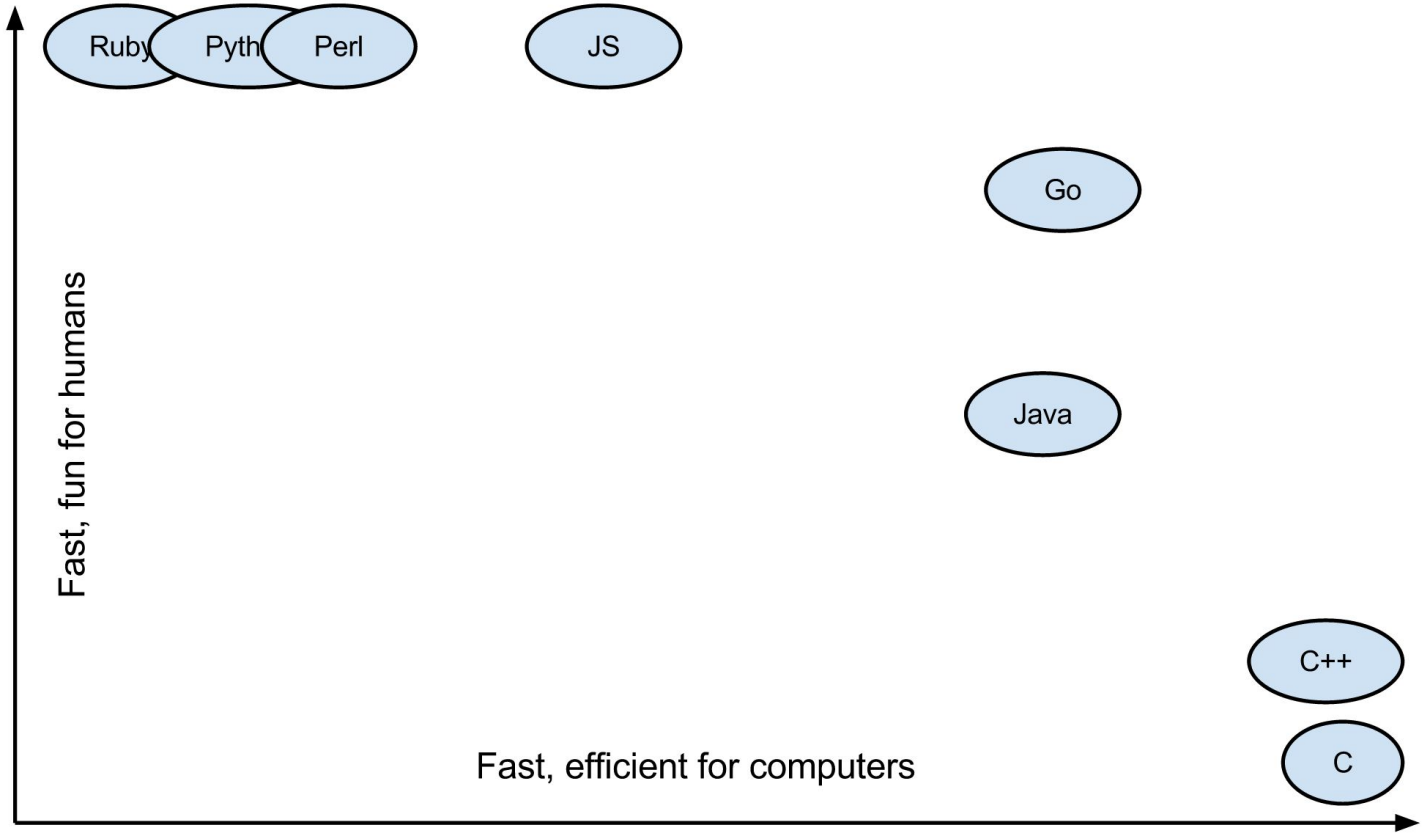
Who made this thing?

- Ken Thompson (B, C, Unix, UTF-8)
- Rob Pike (Unix, UTF-8)
- Robert Griesemer (Hotspot, JVM)
- And a few other engineers at Google

Go was invented by geniuses

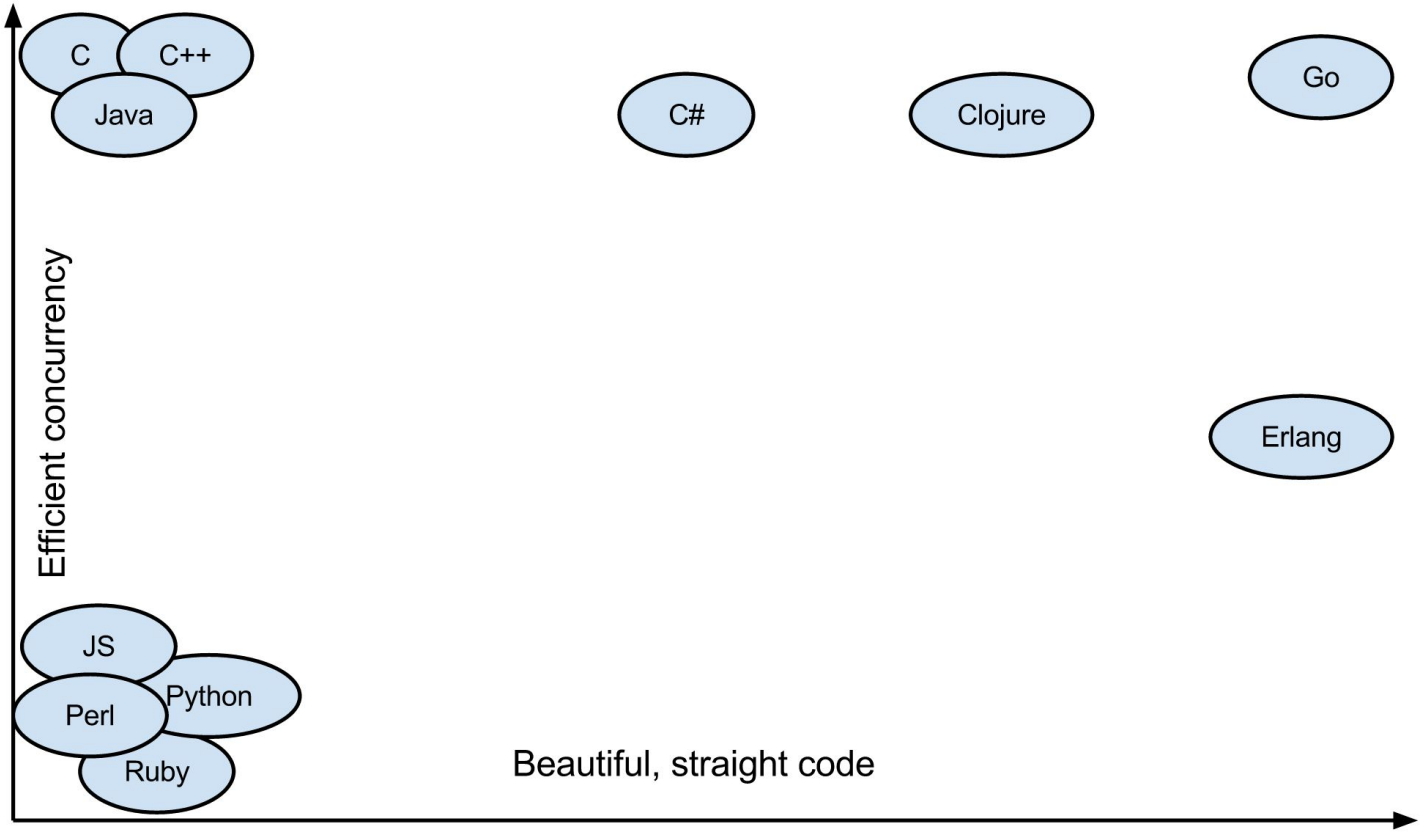
efficient compilation
efficient execution
ease of programming

After Go



source: Brad Fitzpatrick

After Go



source: Brad Fitzpatrick

“Go has its own elegance and programming idioms that make the language productive and fun to code. The language designers set out to create a language that would let them be productive without losing access to the lower-level programming constructs they needed. This balance is achieved through a minimized set of keywords, built-in functions, and syntax. Go also provides a comprehensive standard library. The standard library provides all the core packages programmers need to build real-world web and network based programs.”

~ Bill Kennedy, Go In Action

www.golang-book.com/books/web/01-01

Apps Bookmarks Other Bookmar

Go is a procedural, imperative, modular, object-oriented, statically-typed, garbage-collected, compiled programming language with syntax broadly similar to the C family of languages (C, C++, C#, Java, Javascript, etc...) and CSP-inspired concurrency features. We will examine each of those descriptions in greater detail, but before we do lets write our first Go program.

www.golang-book.com/books/web/01-01

Apps Bookmarks Other Bookmar

Go is a **procedural** programming language because procedures (functions) are stitched together to form a program. It is an **imperative** programming language because functions are built out of statements which are generally phrased as imperative commands.

Here at Google, we believe programming should be fast, productive, and most importantly, fun. That's why we're excited to open source an experimental new language called Go. Go combines the development speed of working in a dynamic language like Python with the performance and safety of a compiled language like C or C++. Typical builds feel instantaneous; even large binaries compile in just a few seconds. And the compiled code runs close to the speed of C. Go lets you move fast.

Go is a great language for systems programming with support for multi-processing, a fresh and lightweight take on object-oriented design, plus some cool features like true closures and reflection.

Want to write a server with thousands of communicating threads? Want to spend less time reading blogs while waiting for builds? Feel like whipping up a prototype of your latest idea? Go is the way to go! Check out the video for more information or visit golang.org.

https://opensource.googleblog.com/2009/11/hey-ho-lets-go.html

Farewell Node.js

https://medium.com/@tjholowaychuk/farewell-node-js-4ba9e7f3e52b

TJ leaving node.js

Posted on July 5, 2014

I just saw the news. TJ Holowaychuk, one of node's important and respected contributors is leaving node.js for Go. Of course, this is not good news, especially for people like me who have invested a lot into node.js and have bet an industrial project on it.

Why is TJ leaving? Part of it has to do with the intrinsic attractiveness of Go. But a large part is related to deficiencies on the node side. Usabililty and the lack of robust error handling come first:

> *Error-handling in Go is superior in my opinion. Node is great in the sense that you have to think about every error, and decide what to do. Node fails however because:*
>
> - *you may get duplicate callbacks*
> - *you may not get a callback at all (lost in limbo)*
> - *you may get out-of-band errors*
> - *emitters may get multiple "error" events*
> - *missing "error" events sends everything to hell*
> - *often unsure what requires "error" handlers*
> - *"error" handlers are very verbose*
> - *callbacks suck*

TJ also complains about APIs, tooling, lack of conventions:

> *Streams are broken, callbacks are not great to work with, errors are vague, tooling is not great, community convention is sort of there, but lacking compared to Go. That being said there are certain tasks which I would probably still use Node for, building web sites, maybe the odd API or prototype. If Node can fix some of its fundamental problems then it has good chance at remaining relevant, but the performance over usability argument doesn't fly when another solution is both more performant and*

TJ Holowaychuk, a prominent module writer, made his farewells to Node.js and is turning to Go, which he calls a “next generation,” programming language in the same company as Rust and Julia.

It’s the latest high profile departure from Node.js to Go, as noted by Zef Hemel in a post today. In particular, he cited Felix Geisendörfer, a frequent contributor who moved on to Go in 2012.

Felix Geisendörfer
@felixge

Follow

bye #nodejs, hello #golang

11:28 AM - 2 Dec 2012

54 24

Hemel also cited Koding, which provides a web-based development environment.