

Getting Started with Go

TechBash 2019

-Mike Rapa

Code from this demo can be found at

<https://github.com/mikerapa/TechBashGoDemo>

About your speaker

Mike Rapa

20+ years in software

B.S. Computer Science @ Thomas Jefferson University

M.E. Software Engineering @ Penn State

Allscripts

Healthcare software and services

10k+ Employees

@mikerapa

github.com/mikerapa



Why Go was created

"The Go programming language was conceived in late 2007 as an answer to some of the problems we were seeing developing software infrastructure at Google. The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created. The problems introduced by multicore processors, networked systems, massive computation clusters, and the web programming model were being worked around rather than addressed head-on. Moreover, the scale has changed: today's server programs comprise tens of millions of lines of code, are worked on by hundreds or even thousands of programmers, and are updated literally every day. To make matters worse, build times, even on large compilation clusters, have stretched to many minutes, even hours.

" -- Rob Pike

Tldr; programming languages born in the 1900s have concurrency as an afterthought and become difficult to manage in large code bases

Google and Go

Go was created by Google engineers, but it's not owned by google

The language and compiler are maintained by the community

Google uses Go for many products

Philosophy

Efficient

Scalable

Productive

Readability is more important than succinctness

- Simple grammar, few keywords
- Slow rate of change
- Minimal syntax overlap
- A junior developer should be able to read an expert developer's code
- Simplicity is a feature

Concurrency is expected

- The concurrency model is prominent in Go
- Concurrency isn't something you try when the performance of your application disappoints

Short compile times

- Delays between coding and running are a loss of productivity

Compiling Go

Build executable with Go Build command

By default, go compiles to 1 file with all of your go code

Go Binaries contain the go runtime (No prerequisites required)

Create binaries for multiple operating systems: Linux, Android, Windows, Free BSD, Mac

The compiler is designed to be as fast as possible

Standard Library

Types

File operations

Network io, web services, http

Testing

Cryptography

Database interfaces

Compression/Decompression

Encoding

Command line interfaces

Math

Go CLI

Create project

Compile

Generate Code

Manage dependencies

Create and run unit tests, report on code coverage

Profiling performance and memory

Code documentation

Composition (not Inheritance)

```
type Person struct {  
    firstName string  
    lastName  string  
}  
  
type Employee struct {  
    Person // Use type without a name for inheritance  
    employeeID int  
}  
  
func main() {  
    employee1 := Employee{employeeID:28, Person:Person{firstName: "Claude", lastName:"Giroux"}}  
    fmt.Printf( format: "Employee %d (%s, %s)", employee1.employeeID, employee1.lastName, employee1.firstName)  
}
```

Data and logic together

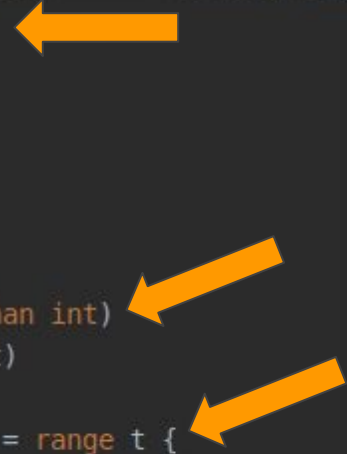
```
type Employee struct {  
    Person // Use type without a name for inheritance  
    employeeID int  
    startDate time.Time  
    startDateString string  
}  
  
func (e *Employee) YearsOfService() (yos float64){  
    startDate, err := time.Parse( layout: "2006-01-02", e.startDateString)  
    if err == nil {  
        yos = time.Since(startDate).Hours()/8760  
    } else {  
        fmt.Println(err)  
    }  
    return  
}  
  
func main() {  
    employee1 := Employee{employeeID:28, Person:Person{firstName: "Claude", lastName:"Giroux"}, startDateString:"2008-12-31"}  
    fmt.Printf( format: "Employee %d (%s, %s)\n", employee1.employeeID, employee1.lastName, employee1.firstName)  
    fmt.Printf( format: "Years of service %f\n", employee1.YearsOfService())  
}
```

Language features

Memory management	Garbage Collector
Classes	No, but structs are like classes in Go
Error handling	Error type. No exceptions. No Try, Catch, Finally.
Typing	Static typing system, compile-time checking
Referencing	By value, pointers
Abstractions	Interfaces
Concurrency	Goroutines, channels
Syntax	A bit like C
Programming Paradigm	Procedural, Optional Object Orientation

Concurrency

```
func GetData(t chan int) {  
    for i:=0; i<3;i ++{  
        time.Sleep(2 * time.Second)  
        fmt.Printf( format: "Adding %d\n", i)  
        t <- i  
    }  
    close(t)  
}  
  
func main() {  
  
    t := make(chan int)  
    go GetData(t)  
  
    for newData:= range t {  
        time.Sleep(3 * time.Second)  
        fmt.Println( a...: "Processing data", newData)  
    }  
  
    fmt.Println( a...: "Done")  
}
```



- Concurrency is a fundamental concept in Go
- Use the keyword `go` to run a routine concurrently. A goroutine is a lightweight thread managed by the Go runtime.
- Channels are used to communicate inputs and outputs to/from goroutines
- The `go` runtime decides how to schedule goroutines upon worker threads

Tooling

Operating system	Architectures	Notes
FreeBSD 10.3 or later	amd64, 386	Debian GNU/kFreeBSD not supported
Linux 2.6.23 or later with glibc	amd64, 386, arm, arm64, s390x, ppc64le	CentOS/RHEL 5.x not supported. Install from source for other libc.
macOS 10.10 or later	amd64	use the clang or gcc [†] that comes with Xcode [‡] for cgo support
Windows 7, Server 2008R2 or later	amd64, 386	use MinGW (386) or MinGW-W64 (amd64) gcc [†] . No need for cygwin or msys.

Development Tools

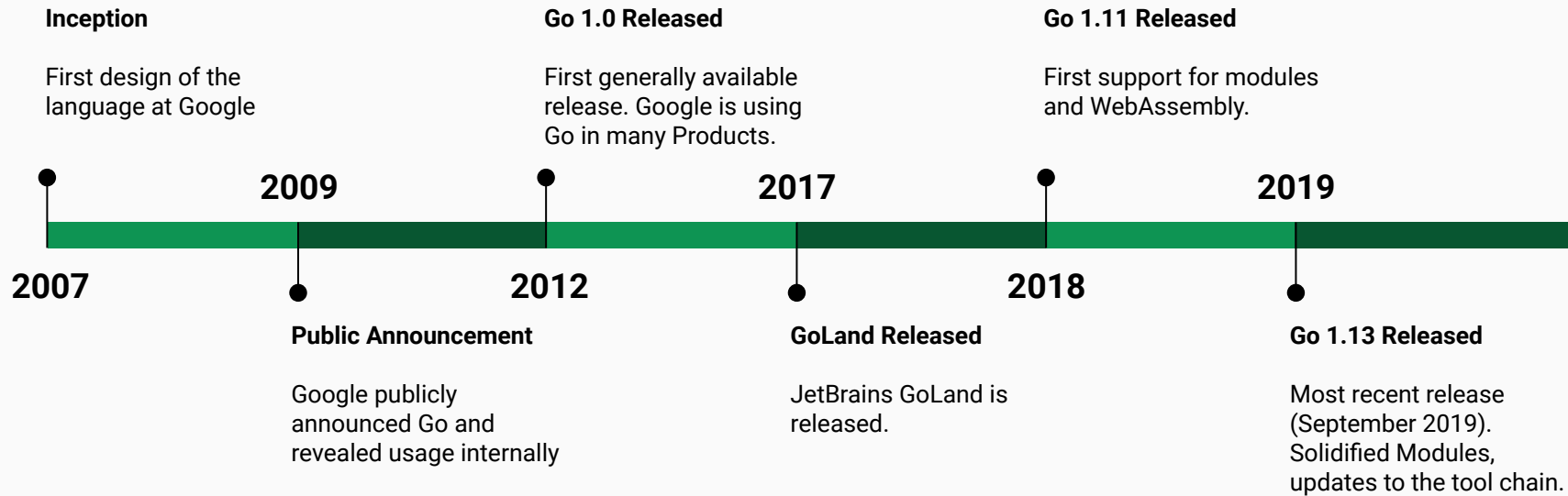


Visual Studio Code with Go Plugin

GoLand - Go specific IDE(JetBrains)

Vim, Atom, Sublime

Timeline



Common usage

Web/Web API

Cloud applications in Azure, Google Cloud and AWS

Small tools (Alternative to PowerShell, Python, etc.)

Kubernetes, Docker, Uber, YouTube, Dropbox, eBay, NetFlix, Twitch, SoundCloud

Questions?