Go Overview

Objectives

- Describe the reasons Go was created
- Explain the design philosophy of Go
- Contrast Go with other languages
- Understand Go basics
- Explore the Go ecosystem
- Locate Go documentation and community resources

Go Defined

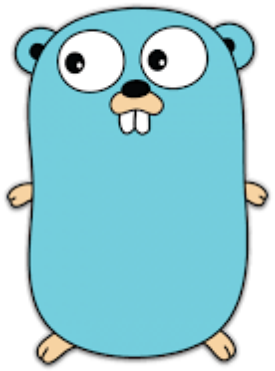
- Go is an open source programming language that makes it easy to build simple, reliable, and efficient software
  -- golang.org

- Often referred to as golang
- Created at Google in 2007
  - Used in some of Google's production systems

- Two major implementations exist
  - Tools are accessed through a single command called go that has a number of subcommands (run, build, get, etc.)
  - gc – Google's Go compiler
    - Supports Linux, OS X, Windows, various BSD/Unix, and mobile devices
    - Self hosted as of v1.5 (the Go compiler is written in Go)
    - The original and principle Go compiler
  - gccgo – A GCC Go frontend
    - Compiles Go with the Gnu C/C++ compiler gcc
    - gccgo tooling has traditionally been slower to compile but faster at runtime due to vast set of optimizations available in gcc
    - Having two compiler implementations ensures the Go spec is concise and explicit

- Go Features
  - Compiled
  - Statically typed
  - C syntax
  - Garbage collection
  - Memory safety features
  - CSP-style concurrency
  - Native UTF-8 strings

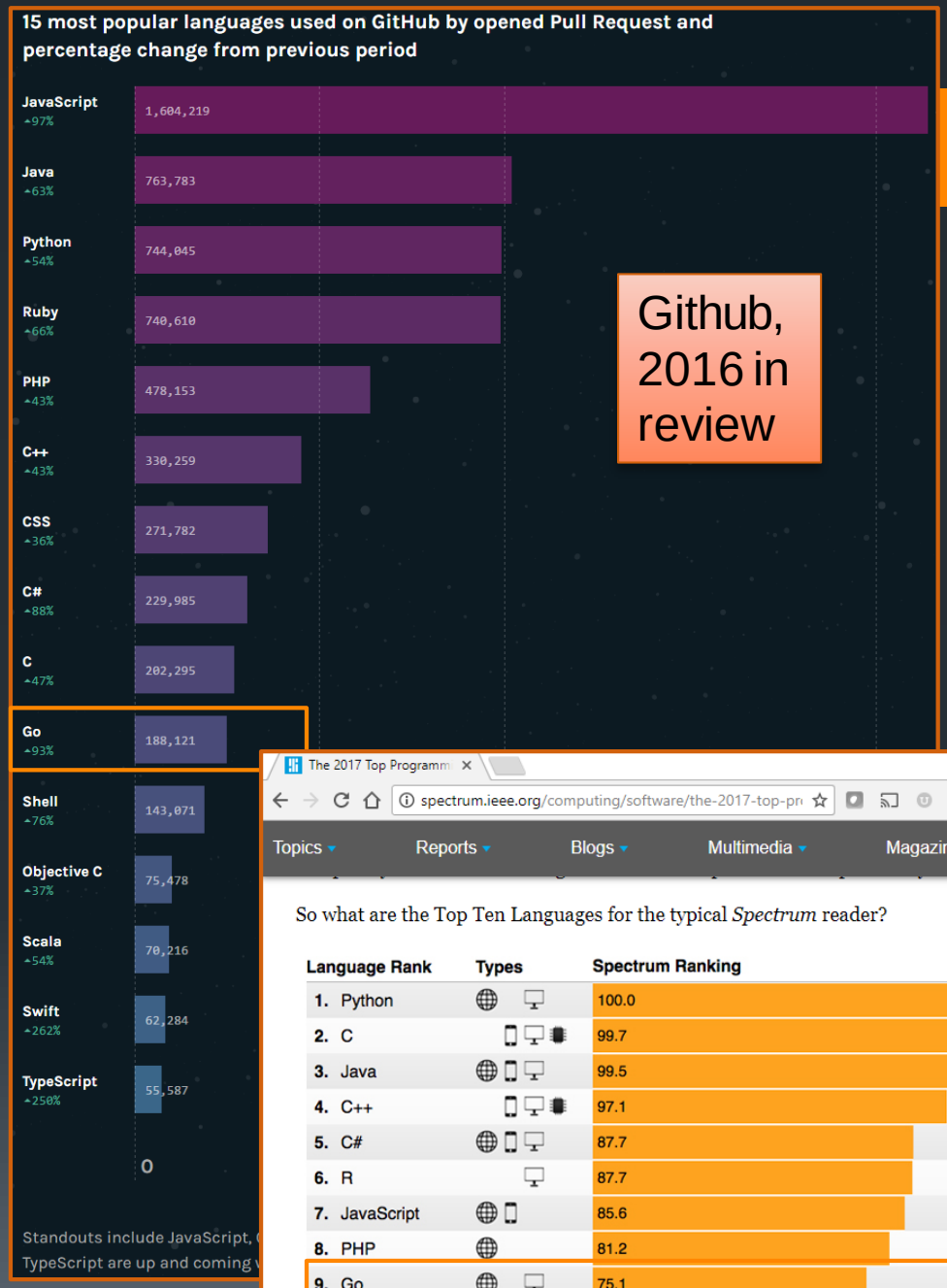
> *The key point here is our programmers are Googlers, they're not researchers. They're typically, fairly young, fresh out of school, probably learned Java, maybe learned C or C++, probably learned Python. They're not capable of understanding a brilliant language but we want to use them to build good software. So, the language that we give them has to be easy for them to understand and easy to adopt.*
>
> *-- Rob Pike*



The Go pher

Motivation for Go

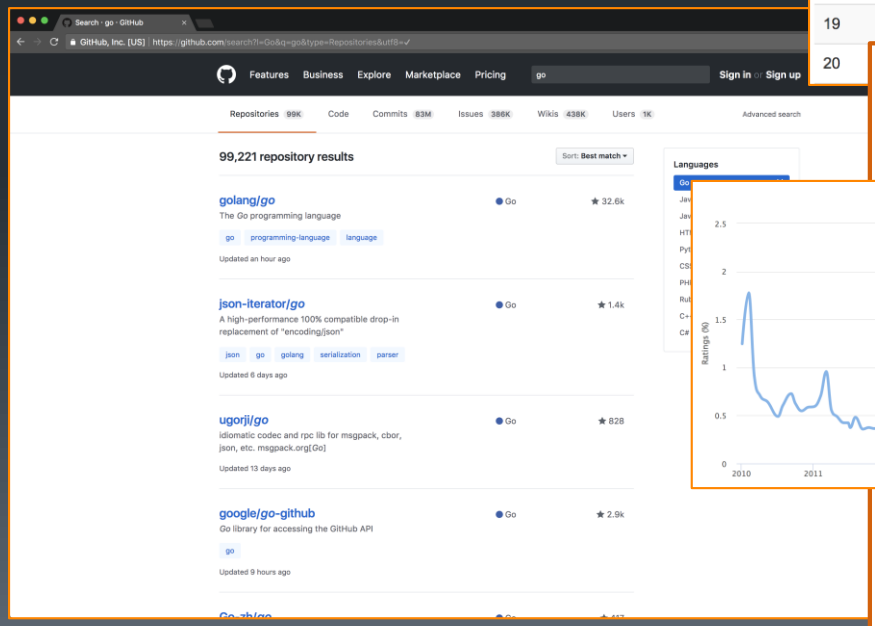
- Don't we have enough programming languages already?
  - Some don't think so
- Go originated as an experiment by Google engineers
  - Robert Griesemer
  - Rob Pike
  - Ken Thompson
- The goal:
  - To design a new programming language that would resolve common criticisms of other languages while maintaining their positive characteristics
- Go included the following features:
  - Static typing
  - Scalable to large systems (like Java and C++)
  - Highly productive and readable
  - No proliferation of mandatory keywords
  - Avoiding repetition
    - "light on the page" like dynamic languages
  - No need for IDEs (integrated development environments), but support for them
  - Support for networking
  - Support for multiprocessing
- In interviews all three designers cited their shared dislike of C++'s complexity as a primary motivation for designing a new language

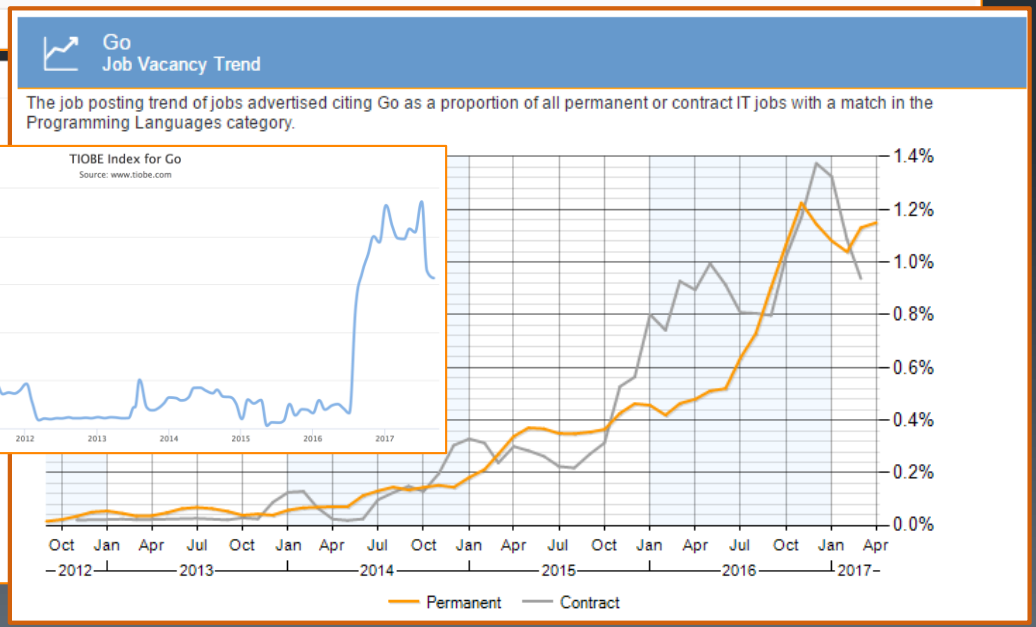


Growing Popularity

- Per the Tiobe index, Go is the 17th most popular programming language as of September 2017
  - Up from 48th the year before
  - Go had the largest place gain of any language in 2009, the year of its introduction
- Nearly 355k repository results match a search on GitHub for Go
  - ~100k marked as Golang language
- Demand for Go programmers has been growing significantly

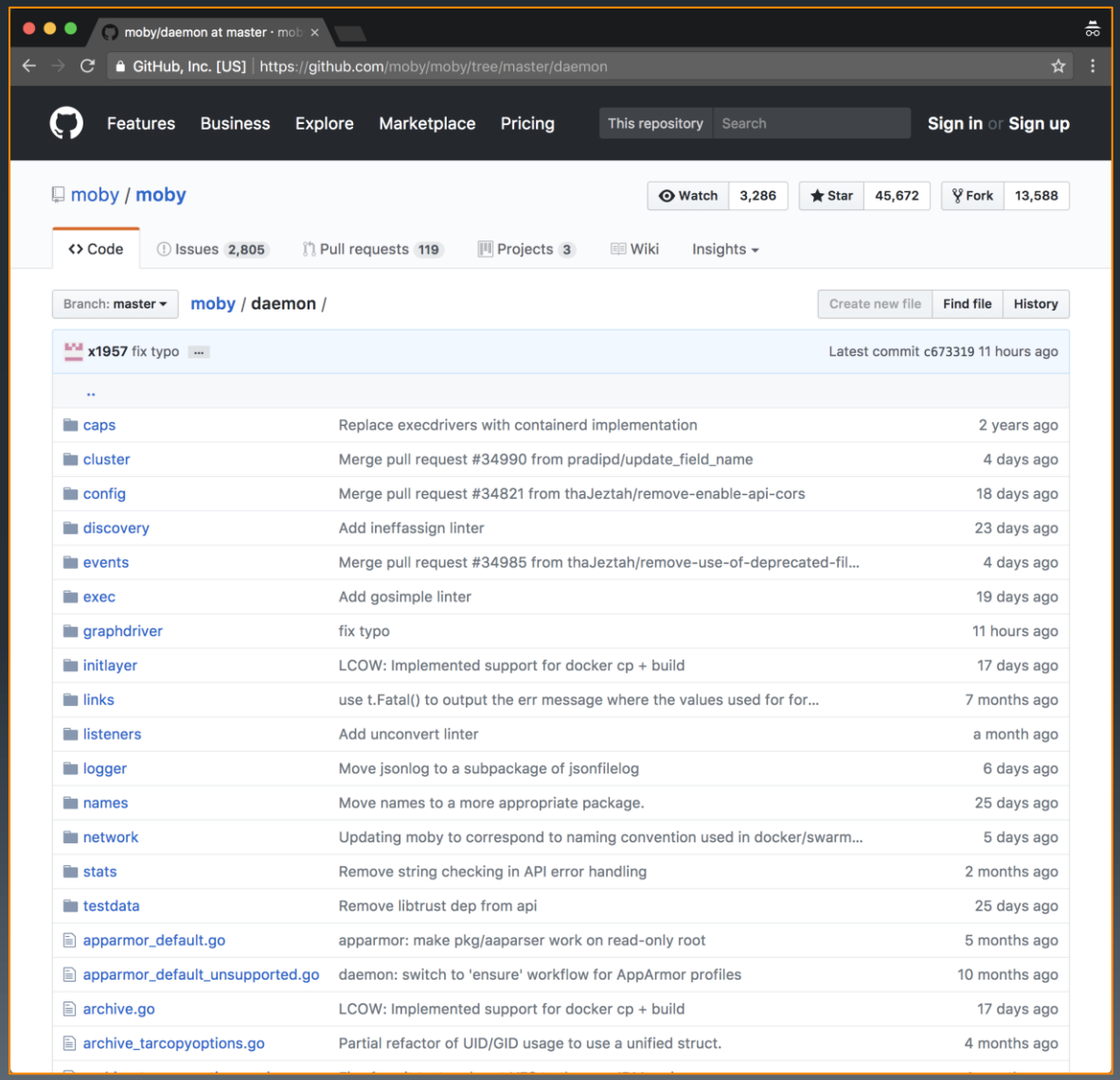
Sep 2017	Sep 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.687%	-5.55%
2	2		C	7.382%	-3.57%
3	3		C++	5.565%	-1.09%
4	4		C#	4.779%	-0.71%
5	5		Python	2.983%	-1.32%
6	7	^	PHP	2.210%	-0.64%
7	6	v	JavaScript	2.017%	-0.91%
8	9	^	Visual Basic .NET	1.982%	-0.36%
9	10	^	Perl	1.952%	-0.38%
10	12	^	Ruby	1.933%	-0.03%
11	18	^^	R	1.816%	+0.13%
12	11	v	Delphi/Object Pascal	1.782%	-0.39%
13	13		Swift	1.765%	-0.17%
14	17	^	Visual Basic	1.751%	-0.01%
15	8	vv	Assembly language	1.639%	-0.78%
16	15	v	MATLAB	1.630%	-0.20%
17	**19**	^	**Go**	**1.567%**	**-0.06%**
18	14	vv	Objective-C	1.509%	-0.34%
19	20	^	PL/SQL	1.484%	+0.04%
20					





The Language of Container Tech

- While Go is used in many areas, container technology is dominated by Go
- Tools written in Go:
  - cAdvisor
  - Consul
  - Docker
  - etcd
  - InfluxDB
  - Kubernetes
  - NATS
  - OCI
    - Containerd
    - Rocket
  - Prometheus
  - …many more…



Go Versions

- 2017-08-24 go1.9 - https://golang.org/doc/go1.9
- 2017-05-24 go1.8.3
- 2017-05-23 go1.8.2
- 2017-04-07 go1.8.1
- 2017-02-16 go1.8
- 2017-01-26 go1.7.5
- 2016-12-01 go1.7.4 & go1.6.4
- 2016-10-19 go1.7.3
- 2016-10-17 go1.7.2
- 2016-09-07 go1.7.1
- 2016-08-15 go1.7
- 2016-07-18 go1.6.3
- 2016-04-19 go1.6.2
- 2016-04-11 go1.6.1 & go1.5.4
- 2016-02-17 go1.6
- 2016-01-13 go1.5.3
- 2015-12-02 go1.5.2
- 2015-09-22 go1.4.3
- 2015-09-08 go1.5.1
- 2015-08-18 go1.5 (Go becomes self hosting)
- 2015-02-17 go1.4.2
- 2015-01-15 go1.4.1
- 2014-12-10 go1.4
- 2014-09-30 go1.3.3
- 2014-09-25 go1.3.2
- 2014-08-12 go1.3.1
- 2014-06-18 go1.3
- 2014-05-05 go1.2.2
- 2014-05-02 go1.2.1
- 2013-11-28 go1.2
- 2013-08-12 go1.1.2
- 2013-06-12 go1.1.1
- 2013-05-13 go1.1
- 2012-09-21 go1.0.3
- 2012-06-13 go1.0.2
- 201204-26 go1.0.1
- 2012-03-28 go1

- Prior to Go v1 maintainers released weekly builds
- In 2011 releases were designated rXX
  - 2011-10-17 release.r58.2 & release.r60.3
  - 2011-10-05 release.r60.2
  - 2011-09-18 release.r60.1
  - 2011-09-07 release.r60
  - 2011-07-31 release.r59
  - 2011-07-12 release.r58.1
  - 2011-06-29 release.r58
  - 2011-06-15 release.r57.2
  - 2011-05-03 release.r57 & release.r57.1
  - 2011-03-06 release.r56
  - … (only weeklies prior to r56)
  - 2009-11-06 weekly.2009-11-06
    - First release on GitHub

Go version 1 (Go 1) defines two things:
- the specification of the language
- the specification of the standard packages

It is intended that programs written to the Go 1 specification will continue to compile and run correctly, unchanged, over the lifetime of that specification

At some indefinite point, a Go 2 specification may arise, but until that time, Go programs that work today should continue to work even as future "point" releases of Go 1 arise

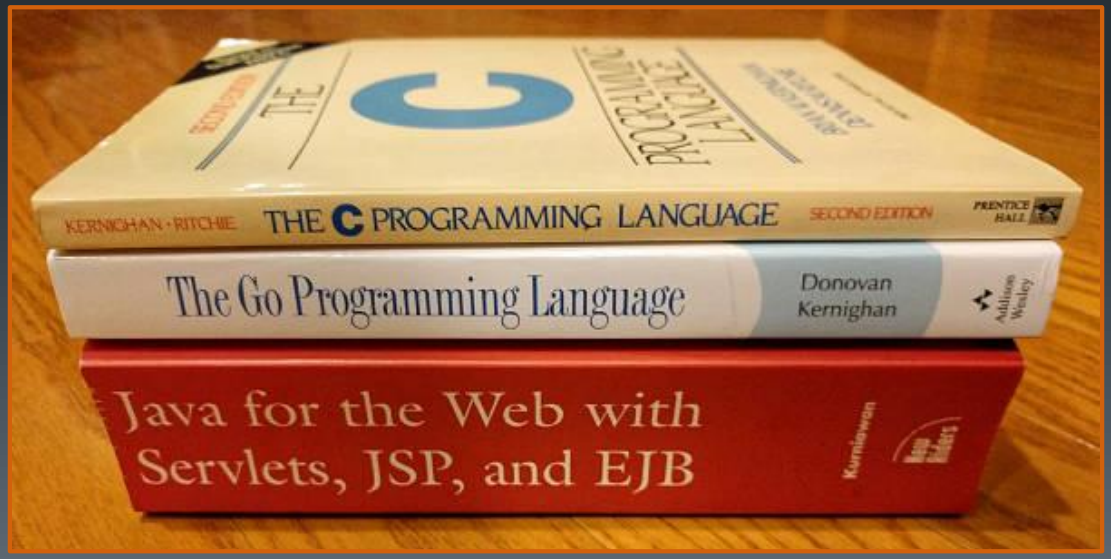
Compatibility is at the source level (binary compatibility for compiled packages is not guaranteed between releases)

APIs may grow, acquiring new packages and features, but not in a way that breaks existing Go 1 code

Go has been post v1.0 for over 5 years
And in use for over 10 years

Go is compact

- K&R C has 32 keywords
- C99 has 37 keywords
- C11 has 44 keywords
- Go has 25 keywords + 18 reserved type and constant names
  - https://golang.org/ref/spec#Keywords
  - https://golang.org/ref/spec#Constants
- Java has 53 keywords
- The current version of C++ has 94 keywords not counting preprocessor directives



High level style comparison

- Go provides an unique mix of features
  - Simple syntax
  - Compiled language performance
  - Static linking runtime simplicity
  - Robust concurrency
  - Absence of OO inheritance
  - A unique approach to interfaces with implicit implementation
  - Memory safety
    - Which can be disabled for low level development
  - A robust set of modern libraries making web and network service development easy

Language comparison

	Python	Ruby	JS/ Node.js	C/C++	Java	Go
semicolons	N	N	Y	Y	Y	N
curly braces	N	N*	Y	Y	Y	Y
static types	N	N	N	Y	Y	Y
easy-to-use concurrency	N	N	Y	N	N	Y
multi-core concurrency	N	N	N	Y	Y	Y
compiled	N	N	N	Y	Y	Y
OO: classes, inheritance	Y	Y	Y	Y	Y	N*

Hello World in Go

- example.go
  - Go source files end with .go
- package main
  - All Go code must exist within a package
  - The first statement in a source file must declare its package
  - The "main" package generates an executable, all other package names generate a library package
- import "fmt"
  - You must import other packages you will use in your code
  - The fmt package is a standard library package that reads and writes formatted text
- func main() { .. }
  - Go programs declare functions with the func keyword
  - The main() function is special when found in the main package and is used to launch an application
  - Function arguments are defined within parenthesis (our example main function receives no arguments)
  - The body of a function is enclosed in curly braces
- fmt.Println("Hello world")
  - fmt.Println() displays parameters to stdout

Welcome | example.go

```go
package main

import "fmt"

func main() {
    fmt.Println("Hellow world")
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
API server listening at: 127.0.0.1:2291
Hellow world
```

Go takes a strong stance on code formatting, declaring a standard format by fiat. This eliminates debate and enables a variety of automated source code transformations and tooling. The gofmt tool rewrites code into the standard format.

Packages

- Go code is organized into packages
  - Similar to libraries or modules in other languages
- A package consists of one or more .go source files in a single directory that define what the package does
- Each source file begins with a package declaration
- Followed by the necessary package imports
  - The Go standard library has over 100 packages
  - You must import exactly the packages you need
    - A program will not compile if there are missing imports or if there are unnecessary ones
- Package main defines an executable program as opposed to a library
- Within package main the function main is where execution of the program begins

```
user@ubuntu:~/go$ cat main.go
package main

import "fmt"

func main() {
  fmt.Println("Hello world!")
}
```

```
package consul

import (
        "errors"
        "fmt"
        "os"
        "testing"
        "time"

        "github.com/hashicorp/consul/consul/structs"
        "github.com/hashicorp/consul/testutil"
        "github.com/hashicorp/net-rpc-msgpackrpc"
        "github.com/hashicorp/serf/serf"
)

```

go

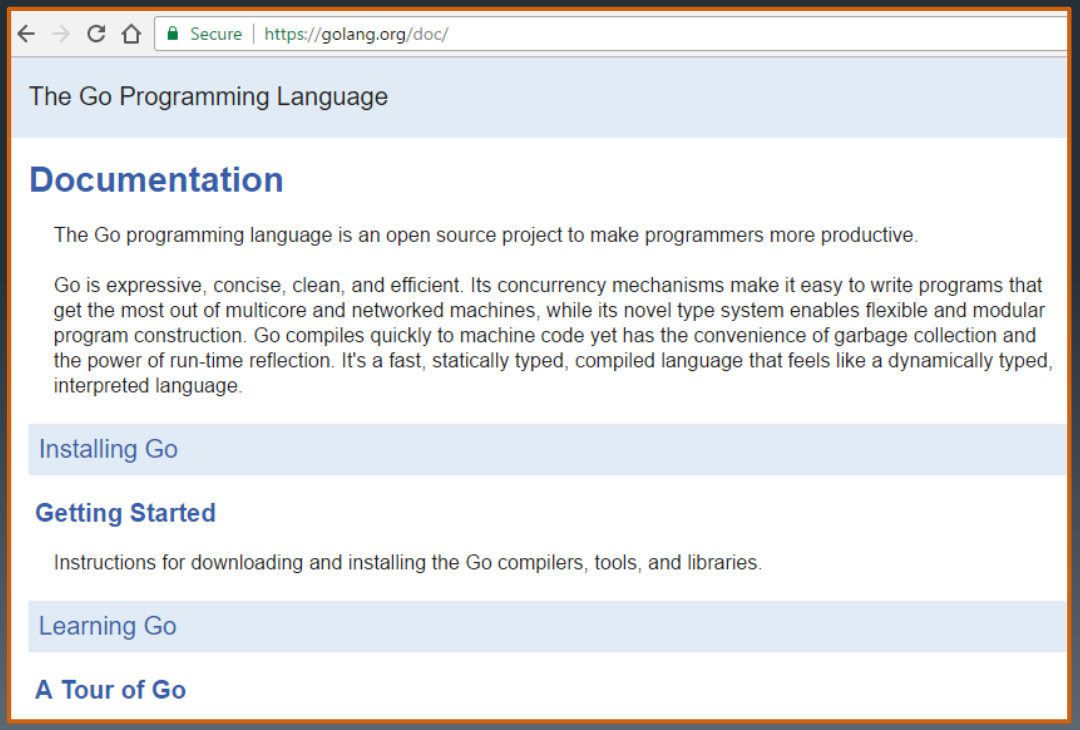
- The go command provides a unified front end to all of the key go tools
  - go run
    - Compiles and runs a program, then cleans the build
  - go build
    - Compiles programs
  - go clean
    - Removes intermediate files
  - go doc
    - Displays the documentation for a package or an exported package feature

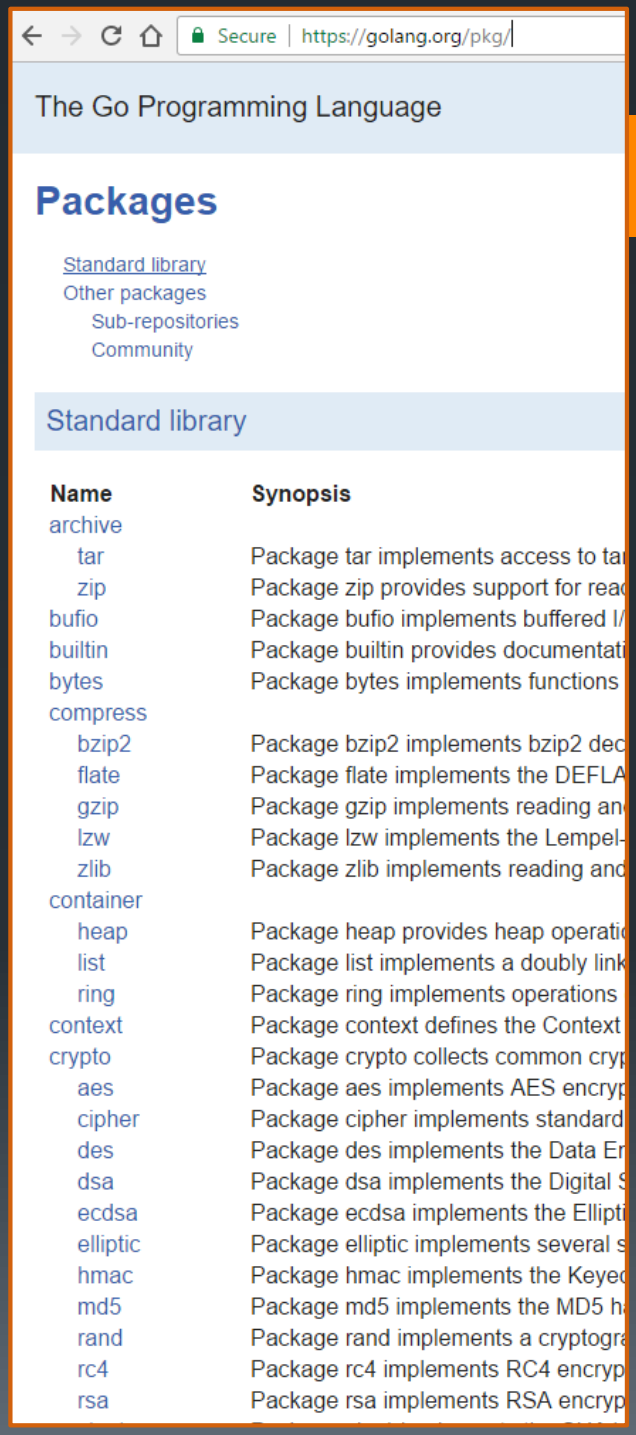
```
user@ubuntu:~$ go
Go is a tool for managing Go source code.

Usage:

        go command [arguments]

The commands are:

        build       compile packages and dependencies
        clean       remove object files
        doc         show documentation for package or symbol
        env         print Go environment information
        bug         start a bug report
        fix         run go tool fix on packages
        fmt         run gofmt on package sources
        generate    generate Go files by processing source
        get         download and install packages and dependencies
        install     compile and install packages and dependencies
        list        list packages
        run         compile and run Go program
        test        test packages
        tool        run specified go tool
        version     print Go version
        vet         run go tool vet on packages

Use "go help [command]" for more information about a command.

Additional help topics:

        c           calling between Go and C
        buildmode   description of build modes
        filetype    file types
                                                        riable

                                                        e lists
                                                        g flags
                                                        g functions

                                                        n about that topic.

user@ubuntu:~$
```

```
user@ubuntu:~/go/lab01$ go doc fmt.Println
func Println(a ...interface{}) (n int, err error)
    Println formats using the default formats for its operands and writes to
    standard output. Spaces are always added between operands and a newline is
    appended. It returns the number of bytes written and any write error
    encountered.
```

Documentation

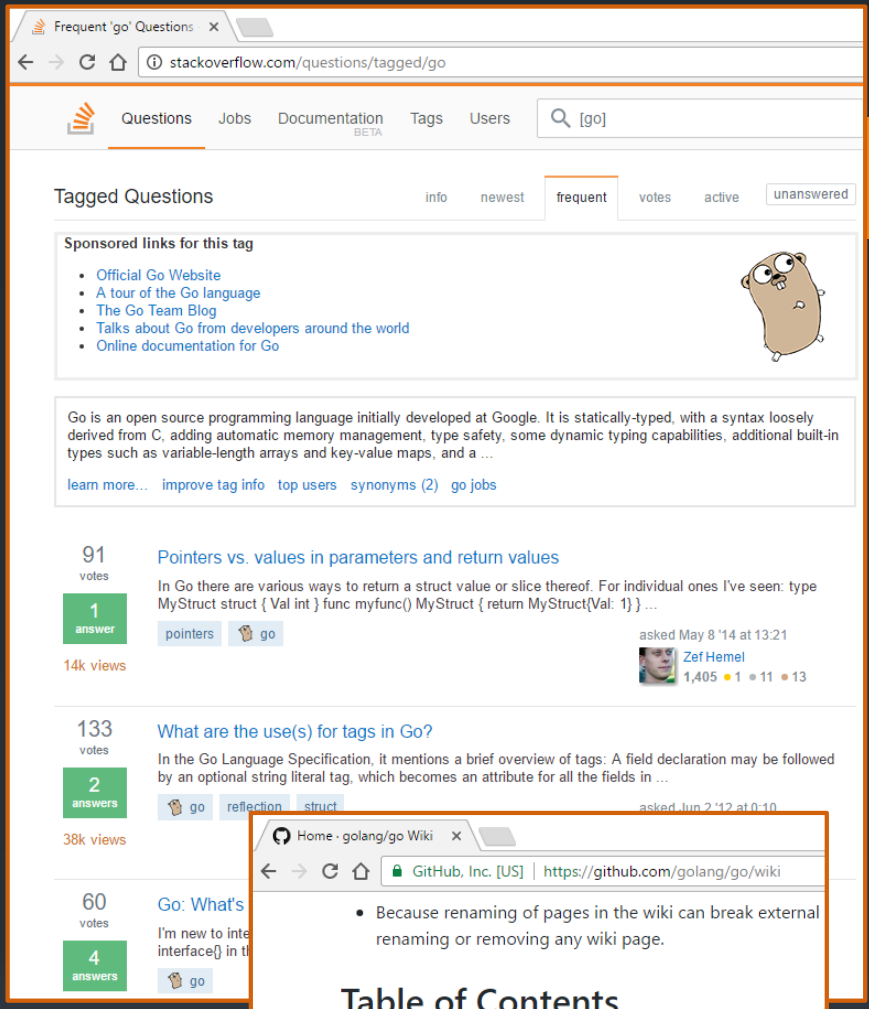
- General Documentation:
  - Getting started, tours, etc.
  - https://golang.org/doc/
- Package Reference:
  - The Go Standard Library contains a wide range of packages
    - > 100 packages
    - You will spend a lot of time here
  - https://golang.org/pkg/

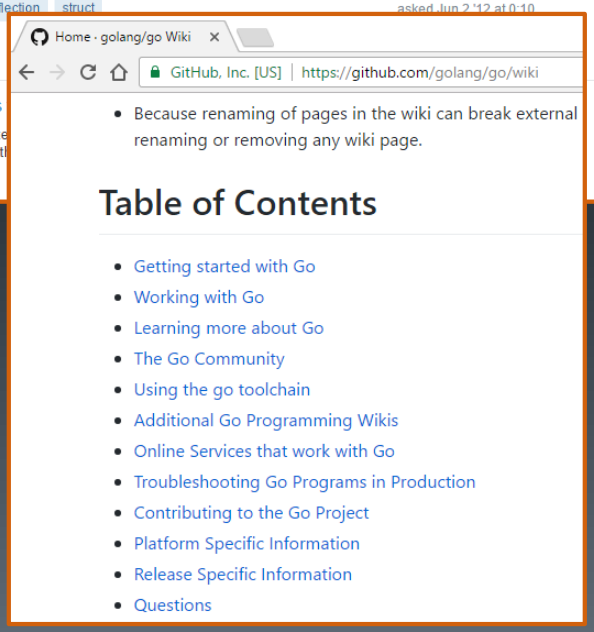




Resources

- Stack Overflow – Go tag
  - Over 17k followers
  - Over 21k questions
- Go Nuts Mailing List - https://groups.google.com/group/golang-nuts
  - Get help from Go users, and share your work on the official mailing list.
  - Search the golang-nuts archives and consult the FAQ and wiki before posting.
- Go Forum - https://forum.golangbridge.org/
  - The Go Forum is a discussion forum for Go programmers.
- Gopher Slack - https://blog.gopheracademy.com/gophers-slack-community/
  - Get live support from other users in the Go slack channel.
- Go IRC Channel - #go-nuts on irc.freenode.net
  - Get live support at #go-nuts on irc.freenode.net, the official Go IRC channel.
- Frequently Asked Questions (FAQ) - https://golang.org/doc/faq
  - Answers to common questions about Go.
- Announcements Mailing List - https://groups.google.com/group/golang-announce
  - Stay informed on the Go announcements mailing list
  - Subscribe to golang-announce for important announcements, such as the availability of new Go releases.
- Go Blog - https://blog.golang.org/
  - The Go project's official blog.
- @golang at Twitter
  - The Go project's official Twitter account.
- Go+ community
  - A Google+ community for Go enthusiasts.
- golang sub-Reddit - https://reddit.com/r/golang
  - The golang sub-Reddit is a place for Go news and discussion.
- Go User Groups - https://golang.org/wiki/GoUserGroups
  - Each month in places around the world, groups of Go programmers ("gophers") meet to talk about Go. Find a chapter near you.
- Go Playground - https://golang.org/play
  - A place to write, run, and share Go code.
- Go Wiki - https://golang.org/wiki
  - A wiki maintained by the Go community.





Go Editors/IDEs

- One of the guiding directives for Go is:
  - No need for integrated development environments, but optional support for them
- A range of editors and IDEs are commonly used with Go
- https://github.com/golang/go/wiki/IDEsAndTextEditorPlugins

- VIM!
  - vim-go
- EMACS
  - You know the type …
- Sublime Text
  - sublime-build plugin for go
- Lime
  - A Sublime lookalike written in Go (!)
  - Not ready for prime time
- Atom
  - go-plus plugin

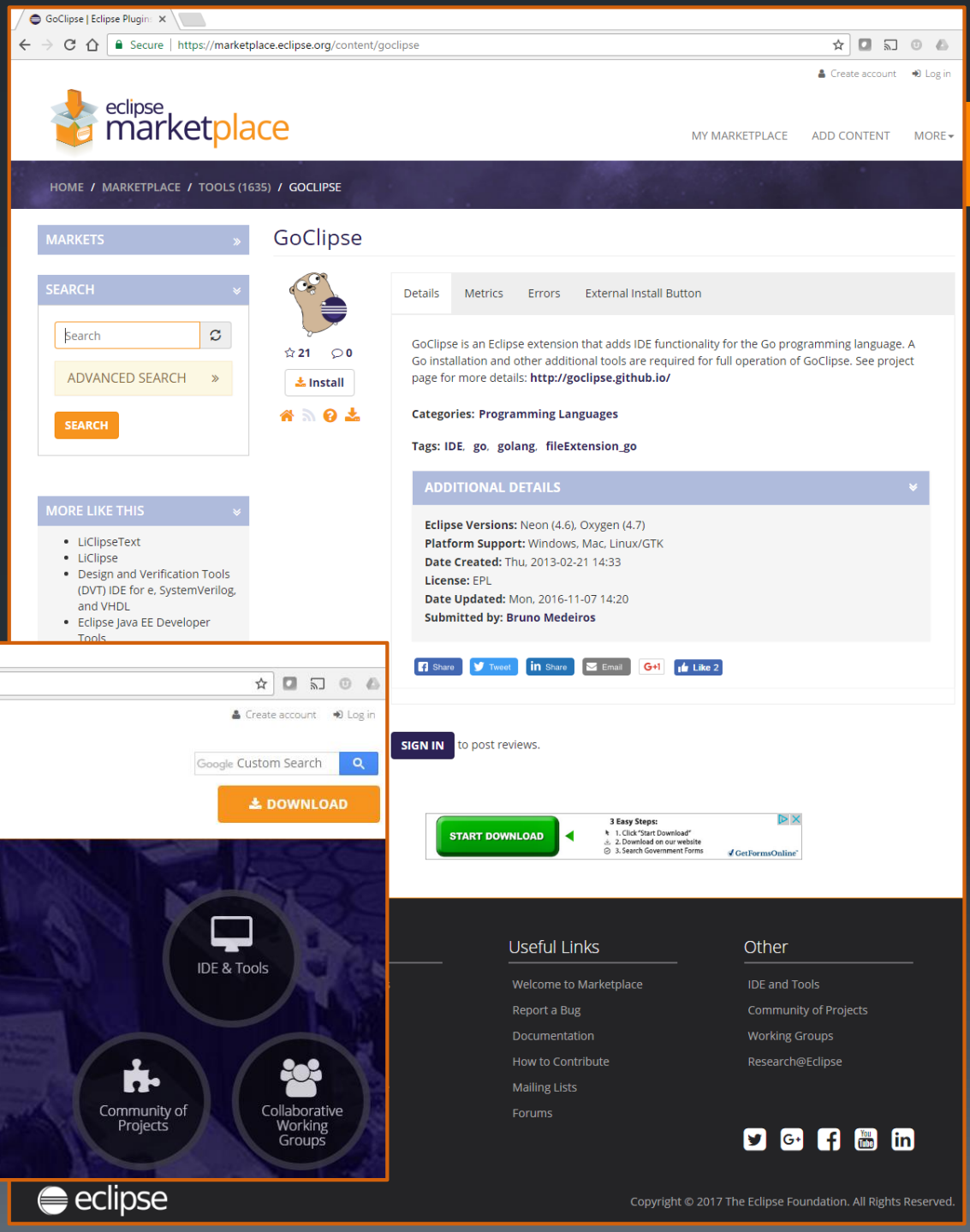
- IDEs with debuggers:
  - Eclipse
    - Goclipse
  - Visual Studio Code
    - vscode-go plugin
  - JetBrains Gogland

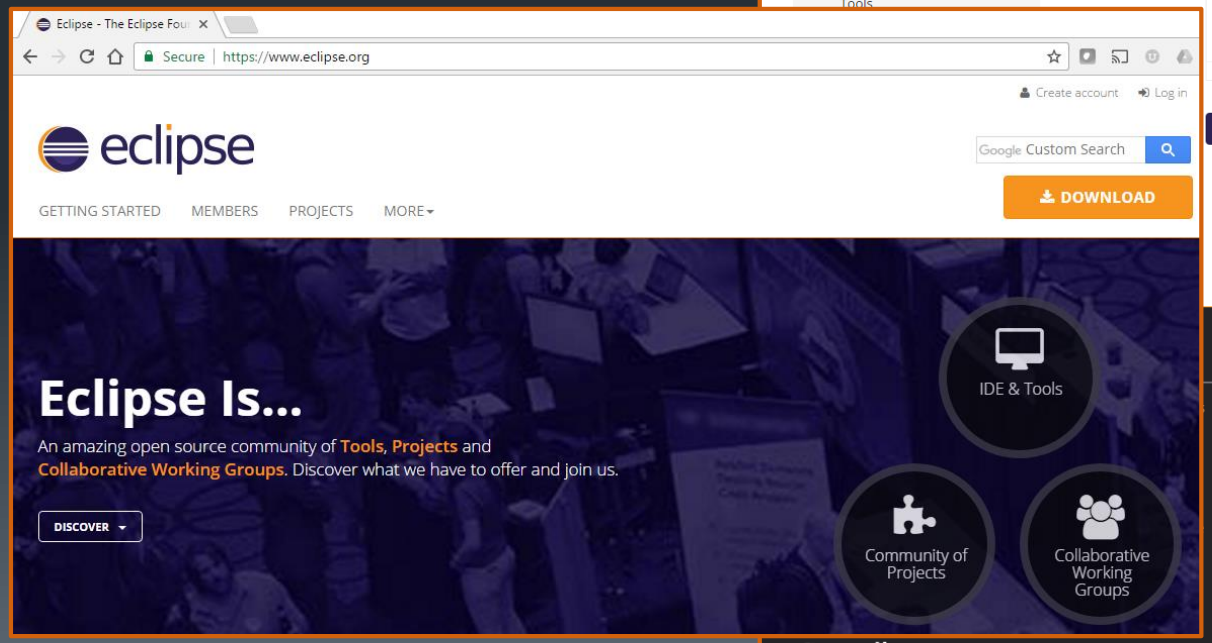
- Common Go editor features
  - code completion
  - automatic imports
  - automatic code formatting
  - linter integration
  - code navigation
- Uncommon features:
  - debugging (not found in straight editor plugins)

- Go tools driving features under the hood
  - gocode: completion lists
  - godoc: signature help
  - godef: Goto definition
  - guru: finding references
  - go-outline: file outlining
  - go-symbols: workspace symbol search
  - gorename: rename identifiers
  - go build and go test: build support
  - golint: linter
  - gometalinter: linter
  - gofmt and goimports and goreturns: formatting
  - gopkgs: import resolution
  - delve: debugger

Eclipse

- Eclipse is a popular cross platform IDE famous for its Java development environment
  - Eclipse is written in Java
- The GoClipse plugin adds support for Go development to Eclipse





Summary

- Go was created to define a language without the key problems of prevalent languages while preserving their strengths
- Go was designed to be simple to use yet highly productive
- Go includes a large standard library empowering important modern features such as web service implementation
- Go programs are divided into packages
- Go eco system has grown tremendously over the last decade
- Go documentation is robust and easy to navigate

Lab: Overview

- Setting up Go tools and creating Go programs
- Optional GoClipse