The Go Language

Goals, Workspaces, Go Type System, CGo, Packaging, Concurrency, Remote Repositories

The Go Language (GoLang) is a next-generation system programming language that is rapidly growing in popularity. Many wildly successful commercial and open-source projects such as Docker and Kubernetes are written in Go. It is appreciated for a range of innovations, such as built-in concurrency including its channel architecture, simplified tooling, sensible conventions, and lots more. This course is aimed at experienced developers and brings them quickly up to speed programming Go and being able to understand and enhance existing Go source trees and being able to build their own.

We cover creating Go commands and libraries, package management, interaction with repositories, structural typing. We look at Go's rich range of system packages. We see how lack of classes and inheritance is not a problem. We explore the use of Go in a range of popular projects and see what real-world benefits it brings to system-level programming.

Go is the language of choice for modern low-level work and it is increasingly being selected by cutting-edge system developers for their most challenging projects. This course helps each attendee become one.

Target Audience

Developers wishing to create modern system-level applications using the Go language.

Prerequisites

Software developers with an object-oriented and system programming (e.g. concurrency) background.

Knowledge of C or other system-level language is a plus.

Notes

All samples and labs in this course use Go v1.12 - the very latest production release.

Our instructor will use the GoLand IDE; attendees may use any suitable editor they wish to select

Contents of One-Day Training Course

Go Introduction

What problem set is Go trying to solve?
A sensible evolution of C
Excellent for system programming
Setting up Go on your dev machine

Tour Of Language Features

Goroutines
Packaging
Building
Functions as first class citizens
Interfaces and structural typing
Lack of classes; no inheritance

Workspace Management

The Go workspace is how code is managed
Naming and file placement conventions
Handling packages
Building & using commands and libraries
Role of `$GOPATH`
`go build` vs. `go install`
Creating / importing packages / exports

Go Type System

bools, strings, ints, runes (codepoints), floats, complex
Zero (default) values
Conversions
Type inferencing
Variables with var, :=, constants

Go Functions

Defining function signatures
Returning more than one result
Named returns
Advanced function usage

Constructs

`if` and `if-else` statements
`for` loop / `defer` statement
`switch` (how break is different)
("C's `while` is spelled `for` in Go" !!)

Grouping of Data Items

Arrays and maps
Slices
Structs and pointers

Types and Their Interfaces

Methods – functions with a receiver arg
Interfaces define sets of methods

Concurrency

Concurrency primitives built into Go itself
goroutines and channels
Synchronization

Testing

Go has a built in testing system
Test preparation
The testing package
Executing tests with `go test`

CGo – Calling C Code

Most OS APIs are written in C
Need to call them and other C libraries
How the pseudo-package C works
How Go and C code can interact

Remote Packages

How to access remote repositories
Use of `go get`
Incorporating remote repositories into app

Project

Review of Go usage in a larger project.