



SEP 20 2023

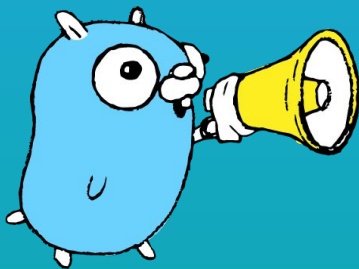
TRANSITIONING TO GO



Robert Pająk

pellared @ GitHub

Splunk



Hello

whoami

01

fundamentals

02

going further

03

more, more

04

q&a

05

whoami



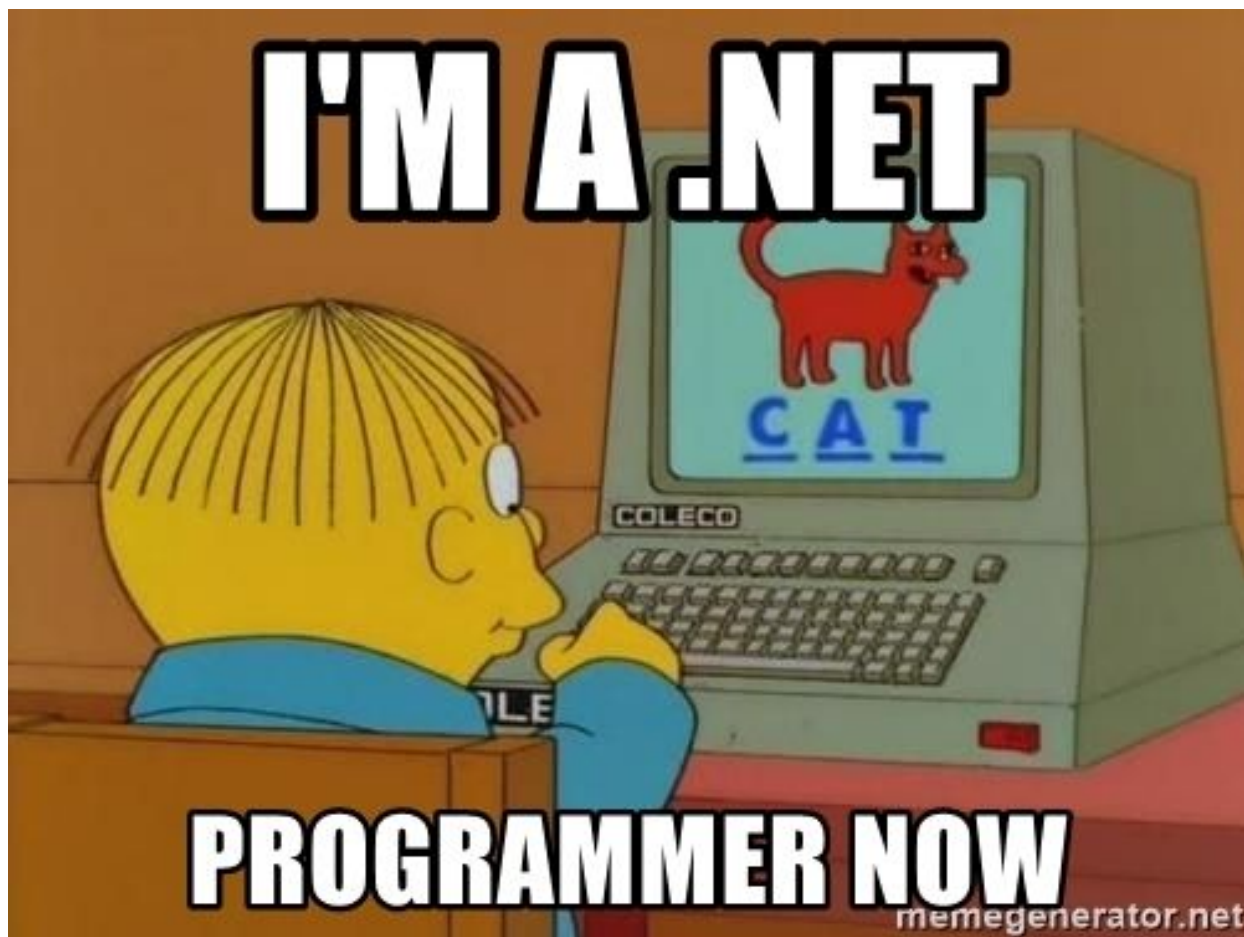
OpenTelemetry Go maintainer



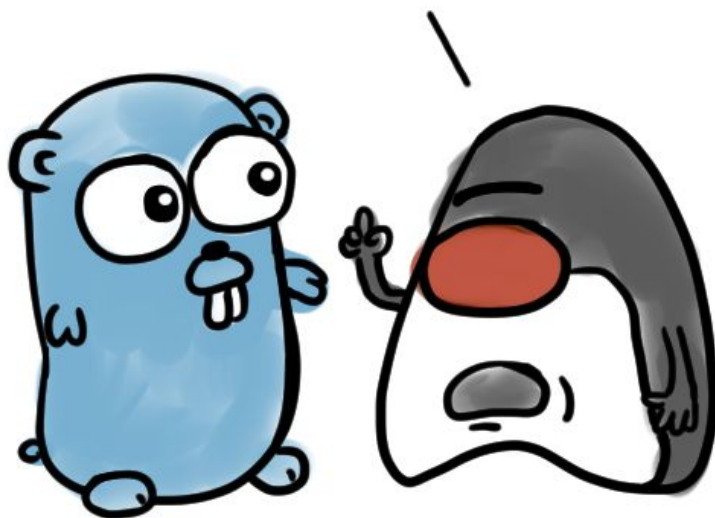
Go language committee member @



Author of [goyek](#)



I had my time, believe me.
Sooner or later they will
call you slow, verbose,
old fashioned...



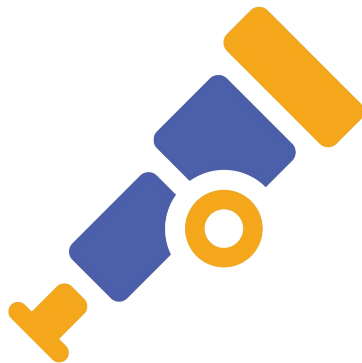
Daniel Stori {turnoff.us}



open source

<https://github.com/golang-templates/seed>

<https://github.com/goyek/goyek>



OpenTelemetry



fundamentals

1. Complete: [Tour of Go](#)
2. Read and try: [How to Write Go Code](#)
3. Read and follow: [Effective Go](#)
4. Read and follow: [CodeReviewComments](#)
5. Check: [Go by Example](#)

守 破 離

Shu

Follow The Rules

Foundational



Ha

Break the Rules




Understanding



Ri

Make the Rules

Mastery

 **A Tour of Go**  

Hello, 世界

Welcome to a tour of the Go programming language.

The tour is divided into a list of modules that you can access by clicking on [A Tour of Go](#) on the top left of the page.

You can also view the table of contents at any time by clicking on the [menu](#) on the top right of the page.

Throughout the tour you will find a series of slides and exercises for you to complete.

You can navigate through them using

- "previous" or [PageUp](#) to go to the previous page,
- "next" or [PageDown](#) to go to the next page.

The tour is interactive. Click the [Run](#) button now (or press [Shift + Enter](#)) to compile and run the program on a remote server. The result is displayed below the code.

These example programs demonstrate different aspects of Go. The programs in the tour are meant to be starting points for your own experimentation.


Edit the program and run it again.

hello.go Imports off Syntax off

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hello, 世界")
7 }
8
```

Reset Format Run

< 1/5 >



How to Write Go Code

Table of Contents

Introduction

Code organization

Your first program

 Importing packages from your module

 Importing packages from remote modules

Testing

What's next

Getting help

Effective Go

Table of Contents

Introduction	Constants
Examples	Variables
Formatting	The init function
Commentary	Methods
Names	Pointers vs. Values
Package names	Interfaces and other types
Getters	Interfaces
Interface names	Conversions
MixedCaps	Interface conversions and type assertions
Semicolons	Generality
Control structures	Interfaces and methods
If	The blank identifier
Redeclaration and reassignment	The blank identifier in multiple assignment
For	Unused imports and variables
Switch	Import for side effect
Type switch	Interface checks
Functions	Embedding
Multiple return values	Concurrency
Named result parameters	Share by communicating
Defer	Goroutines
Data	Channels
Allocation with new	Channels of channels
Constructors and composite literals	Parallelization
Allocation with make	A leaky buffer
Arrays	Errors
Slices	Panic
Two-dimensional slices	Recover
Maps	A web server
Printing	
Append	
Initialization	

Go Code Review Comments

This page collects common comments made during reviews of Go code, so that a single detailed explanation can be referred to by shorthands. This is a laundry list of common style issues, not a comprehensive style guide.

You can view this as a supplement to [Effective Go](#).

Additional comments related to testing can be found at [Go Test Comments](#)

Google has published a longer [Go Style Guide](#).

Please **discuss changes** before editing this page, even *minor* ones. Many people have opinions and this is not the place for edit wars.

Go by Example: Hello World

Our first program will print the classic “hello world” message. Here’s the full source code.

```
package main

import "fmt"

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



To run the program, put the code in `hello-world.go` and use `go run`.

```
$ go run hello-world.go
hello world
```

Sometimes we’ll want to build our programs into binaries. We can do this using `go build`.

```
$ go build hello-world.go
$ ls
hello-world  hello-world.go
```

We can then execute the built binary directly.

```
$ ./hello-world
hello world
```

Now that we can run and build basic Go programs, let’s learn more about the language.



going further

The Go Programming Language

Alan A. A. Donovan
Brian W. Kernighan



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

The Go Memory Model

<https://exercism.org>

{ } exercism

Learn

Discover

Contribute

More



67 languages for you to learn

Become fluent in your chosen programming languages by following the tracks created by our awesome team of contributors

Go

Go

Learning Mode



140 exercises



34 concepts

Procedural

Imperative

Static

Meet your mentor

bitfield

51.3k

John Arundel

Professional Go trainer and author of 'For the Love of Go', and other books I won't weary you by enumerating.



Iteration 1 was submitted

4y ago



bitfield

4y ago

Great job!

- You could make the function more readable by inverting the condition of the `if` statement handling the error case. The rest of the function can then be outdented because an `else` branch is not needed. A Go proverb says "The happy path is left-aligned", meaning the normal flow of control follows the left-hand margin, and only exceptional or error cases are indented.
- Did you know that a map in Go never panics if you ask it for a nonexistent key? Instead it returns the **zero value** for that type. For example, if you have a `map[X]bool` it will return `false` if the entry does not exist. So you can query the `map` without using the `value, ok` syntax: `if somemap[somekey] {...}`.

Try a map of `bool` here, so that you can write:

```
if seen[letter] {
```



Iteration 2 was submitted

4y ago



bitfield

4y ago

What I mean by inverting the `if` statement is this:

```
if !unicode.IsLetter(char) {  
    continue
```

Learn Go with Tests

⋮



Go 101

(an up-to-date knowledge base for Go programming self learning)

- = Go (Fundamentals) 101 - =

- = Go Generics 101 - =

- = Go Optimizations 101 - =

- = Go Details & Tips 101 - =

- = Go Quizzes 101 - =

- = Go 101 Apps & Libs - =

- = Go 101 Blog - =

The Go Blog

Govulncheck v1.0.0 is released!, 13 July 2023

Julie Qiu, for the Go security team

Version v1.0.0 of golang.org/x/vuln has been released, introducing a new API and other improvements.

Go 1.21 Release Candidate, 21 June 2023

Eli Bendersky, on behalf of the Go team

Go 1.21 RC brings language improvements, new standard library packages, PGO GA, backward and forward compatibility in the toolchain and faster builds.

Go Developer Survey 2023 Q1 Results, 11 May 2023

Alice Merrick

An analysis of the results from the 2023 Q1 Go Developer Survey.

Code coverage for Go integration tests, 8 March 2023

Than McIntosh

Code coverage for integration tests, available in Go 1.20.



more, more

<https://go.dev/learn/>



Why Go ▼

Learn

Docs ▼

Packages

Community ▼

Gopher Reading List

中文版

Here is a reading list of blog posts about [Go](#). It aspires to include only the most useful and relevant material that anyone writing Go should eventually read. By definition, the list is a work in progress.

Rather than being comprehensive, the list is a curated selection fixed at 200 entries.

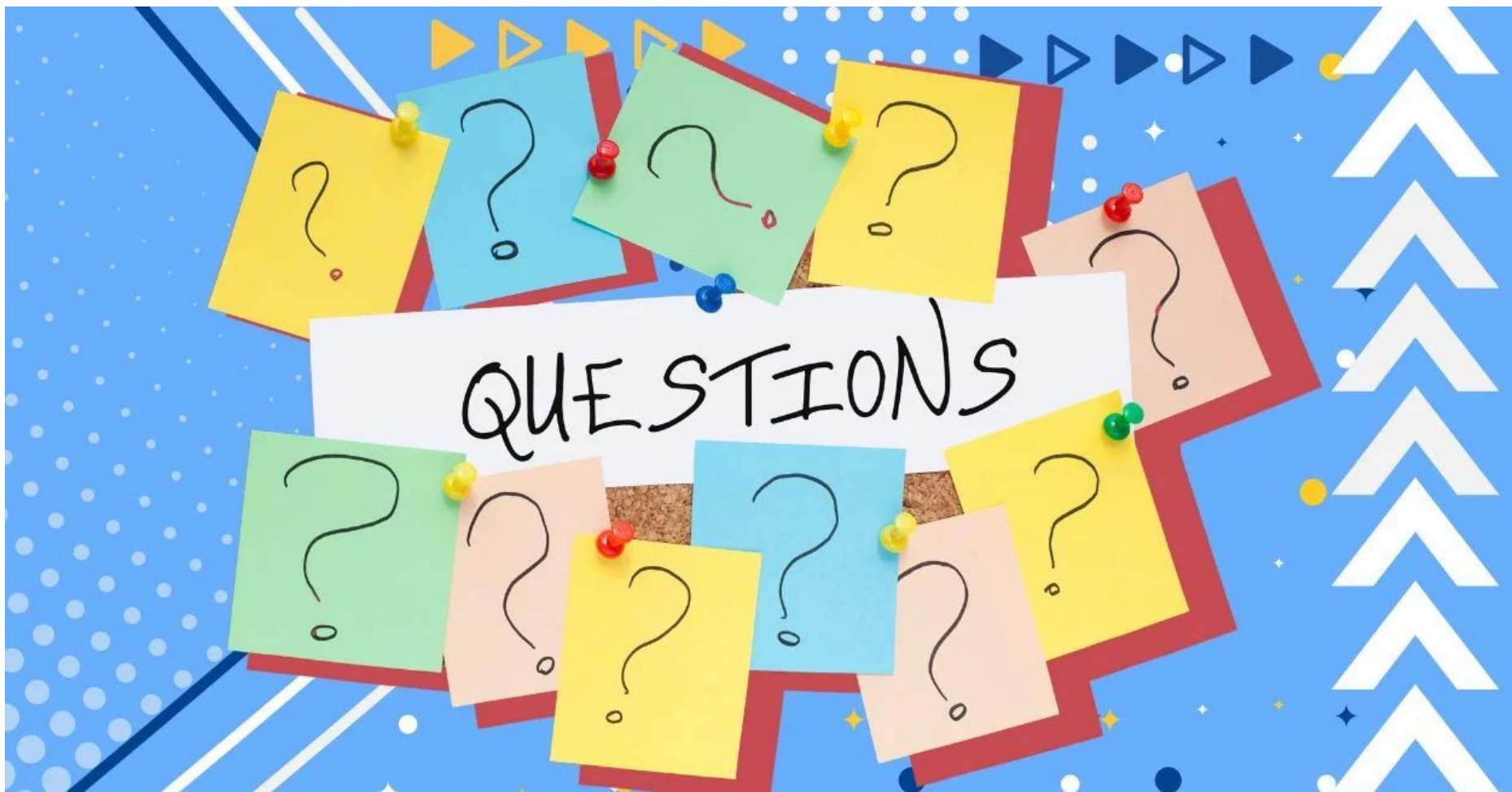
Go is growing fast and so are the number of blog posts about it. If an interested reader knows of a great post not on this list, please open an issue with a link to the post. Not every blog post linked in an issue will make its way into the list. Nonetheless, the [issue list](#) (both open and closed) is a good source of additional reading material.

NOTE: Any new additions will need to replace something else on the list to keep it at a fixed length.

1. Study <https://github.com/golang/go>
2. Check the code of the packages that you are using

- CLI tool (e.g. curl)
- Rewriting scripts (e.g. bash scripts)
- Automated tests (e.g. load tests, REST API tests)
- Network service (e.g. blog portal)

Side note: Consider finding a mentor.



<https://invite.slack.golangbridge.org/>



slack