A Whirlwind Tour of Go Just the Cool Parts

Steve Willoughby

10-Apr-2024

v0.0.1



The Point

- "What is Go?"
- "What is it actually good for?"
- "Why should I care?"

- Designed by Rob Pike, Ken Thompson, and Robert Griesemer.
- Includes direct experience with C from day 1 to now.



- Designed by Rob Pike, Ken Thompson, and Robert Griesemer.
- Includes direct experience with C from day 1 to now.
- "If we were to design C today, knowing what we know now, on today's technology..."

- Designed by Rob Pike, Ken Thompson, and Robert Griesemer.
- Includes direct experience with C from day 1 to now.
- "If we were to design C today, knowing what we know now, on today's technology..."
 - ∴ Go's syntax is very much like C's
 - ... but cleaned up and streamlined a bit.



- Designed by Rob Pike, Ken Thompson, and Robert Griesemer.
- Includes direct experience with C from day 1 to now.
- "If we were to design C today, knowing what we know now, on today's technology..."
 - ∴ Go's syntax is very much like C's
 - ... but cleaned up and streamlined a bit.
- Dreamed up while waiting on a 45-minute C++ compile



- Designed by Rob Pike, Ken Thompson, and Robert Griesemer.
- Includes direct experience with C from day 1 to now.
- "If we were to design C today, knowing what we know now, on today's technology..."
 - .: Go's syntax is very much like C's
 - ... but cleaned up and streamlined a bit.
- Dreamed up while waiting on a 45-minute C++ compile
 - Fast compilation
 - Native binary compiler with low overhead
 - Strong static typing
 - Extraordinarily spartan



Go Syntax

• Type declarations follow identifier names

```
var x int
var UserName string
func AddNumbers(x, y int) int { ... }
func DivideNumbers(x, y int) (int, error) { ... }
type Shape struct {
   X
         int
        int
   Color ColorCode
```

• Basic unit is a package (namespace boundary).

- Basic unit is a package (namespace boundary).
- Multiple source files in a package, in the same directory tree.

- Basic unit is a *package* (namespace boundary).
- Multiple source files in a package, in the same directory tree.
- Every program must have a main package.
- The main package has a main function.

- Basic unit is a package (namespace boundary).
- Multiple source files in a package, in the same directory tree.
- Every program must have a main package.
- The main package has a main function.
- Import packages into the program using the import statement.
- Always prefix identifiers from imported packages with their package name.

- Basic unit is a package (namespace boundary).
- Multiple source files in a package, in the same directory tree.
- Every program must have a main package.
- The main package has a main function.
- Import packages into the program using the import statement.
- Always prefix identifiers from imported packages with their package name.
- Identifiers can be *public* or *private* w/r/t package boundaries.
 - Identifier names starting with an uppercase letter are public.
 - All others are private.

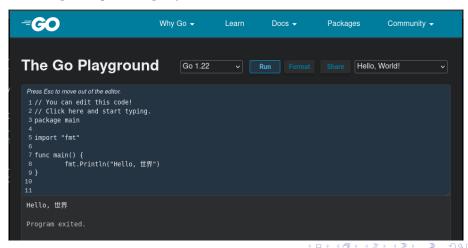


Hello, World

```
/* Standard-issue "Hello, World" program in Go */
package main
import "fmt"
func main() {
     fmt.Println("Hello, 世界")
}
```

The Playground

- Interactive playground to immediately try something in Go.
- https://go.dev/play/



Importing Third-Party Packages

Standard library package names are simple names:

```
import "fmt"
import "encoding/json"
import "flag"
import "math"
```

Importing Third-Party Packages

• Standard library package names are simple names:

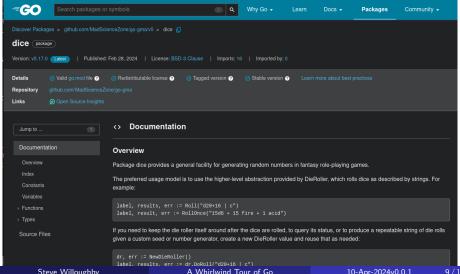
```
import "fmt"
import "encoding/json"
import "flag"
import "math"
```

Getting packages from public repositories:

```
import "github.com/MadScienceZone/go-gma/v5/dice"
```

Automatic API Documentation

• https://pkg.go.dev/repository-url



```
import "fmt"
import "encoding/json"
import "flag"
import "math"
```

```
import "fmt"
import "encoding/json"
import "flag"
import "math"
import (
   "fmt"
   "encoding/json"
   "flag"
   "math"
```

```
initialized bool
var userNames
                 []string
var Greeting
                 string
                           = "Hello"
var TheAnswer
                           = 42
var
    initialized bool
    userNames
                 []string
    Greeting
                 string
                          = "Hello"
    TheAnswer
                           = 42
```

```
const initialized = false
const Greeting = "Hello"
const TheAnswer byte = 42

const (
   initialized = false
   Greeting = "Hello"
   TheAnswer byte = 42
)
```

```
type MessageType byte
const (
    ServerCommand MessageType = 0
    ServerReply MessageType = 1
    ServerError MessageType = 2
    UrgentMessage MessageType = 3
)
```

```
type MessageType byte
const (
    ServerCommand MessageType = 0
    ServerReply
                 MessageType = 1
    ServerError MessageType = 2
   UrgentMessage MessageType = 3
type MessageType byte
const (
    ServerCommand MessageType = iota
    ServerReply
                  MessageType = iota
    ServerError
                  MessageType = iota
    UrgentMessage MessageType = iota
```

```
type MessageType byte
const (
    ServerCommand MessageType = 0
    ServerReply MessageType = 1
    ServerError MessageType = 2
    UrgentMessage MessageType = 3
type MessageType byte
const (
    ServerCommand MessageType = iota
    ServerReply
    ServerError
    UrgentMessage
```

type MessageType byte

```
const (
    ServerCommand MessageType = 0
    ServerReply MessageType = 1
    ServerError MessageType = 2
    UrgentMessage MessageType = 3
type MessageType byte
const (
    ServerCommand MessageType = 1 << iota
    ServerReply
    ServerError
    UrgentMessage
```