

Golang in Practices







The Origins of Go



— — —
Hello, **World!**



Hello, World!

Create Directory call hello world:

1. mkdir helloworld

Create file call main.go:

1. touch main.go
2. copy and paste the code in main.go

Run helloworld app:

1. go run main.go

Compile and Run helloworld app:

1. go build
2. ./helloworld

```
package main
```

```
import "fmt"
```

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



Hello, **World!**



Keywords & Syntax

Standard Library & Go Doc

```
package main
```

```
import "fmt"
```

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

| | | | | |
|----------|-------------|--------|-----------|--------|
| break | default | func | interface | select |
| case | defer | go | map | struct |
| chan | else | goto | package | switch |
| const | fallthrough | if | range | type |
| continue | for | import | return | var |

ref: <https://golang.org/ref/spec#Keywords>





Keywords & Syntax

Standard Library & Go Doc



Version: go1.16.6 **LATEST** | Published: Jul 12, 2021 | License: BSD-3-Clause | Imports: 9 | Imported by: 1944033

Jump to ...

Documentation

Overview

▸ Index

Constants

Variables

▸ Functions

▸ Types

Source Files

Index

`func Errorf(format string, a ...interface{}) error`
`func Fprint(w io.Writer, a ...interface{}) (n int, err error)`
`func Fprintf(w io.Writer, format string, a ...interface{}) (n int, err error)`
`func Fprintln(w io.Writer, a ...interface{}) (n int, err error)`
`func Fscan(r io.Reader, a ...interface{}) (n int, err error)`
`func Fscanf(r io.Reader, format string, a ...interface{}) (n int, err error)`
`func Fscanln(r io.Reader, a ...interface{}) (n int, err error)`
`func Print(a ...interface{}) (n int, err error)`
`func Printf(format string, a ...interface{}) (n int, err error)`
`func Println(a ...interface{}) (n int, err error)`
`func Scan(a ...interface{}) (n int, err error)`
`func Scanf(format string, a ...interface{}) (n int, err error)`
`func Scanln(a ...interface{}) (n int, err error)`
`func Sprint(a ...interface{}) string`
`func Sprintf(format string, a ...interface{}) string`
`func Sprintln(a ...interface{}) string`
`func Sscan(str string, a ...interface{}) (n int, err error)`
`func Sscanf(str string, format string, a ...interface{}) (n int, err error)`
`func Sscanln(str string, a ...interface{}) (n int, err error)`
`type Formatter`
`type GoStringer`
`type ScanState`
`type Scanner`
`type State`
`type Stringer`

ref: <https://pkg.go.dev/fmt@go1.16.6>



Go Commands

```
go run main.go
```

```
go build  
./helloworld
```

```
go build -o app  
./app
```

```
go doc fmt.Println
```

```
go env
```

```
godoc
```





The Origins of Go

Go was conceived in September 2007 by Robert Griesemer, Rob Pike, and Ken Thompson, all at Google, and was announced in November 2009.

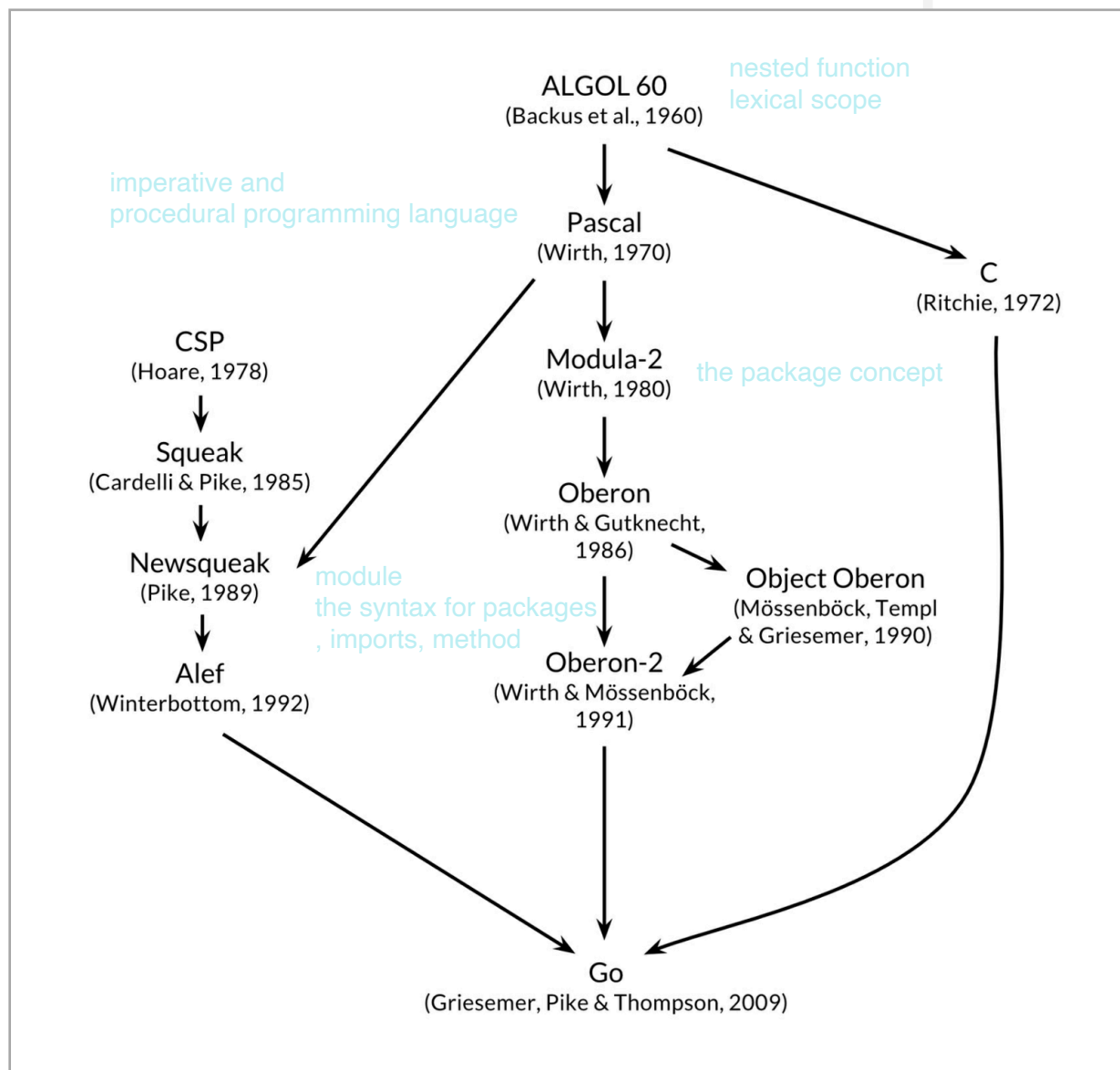
The goals of the language and its accompanying tools were to be **expressive, efficient** in both **compilation** and **execution**, and **effective in writing reliable and robust programs**.



ref: Donovan, Alan A. A.; Kernighan, Brian W.. The Go Programming Language



The Origins of Go



ref: Donovan, Alan A. A.; Kernighan, Brian W.. The Go Programming Language

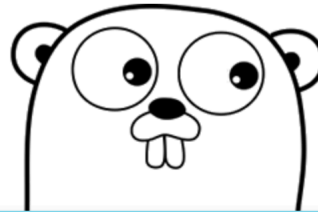




Go Philosophy

[Documents](#)[Packages](#)[The Project](#)[Help](#)[Blog](#)[Play](#)

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

[Download Go](#)

Binary distributions available for Linux, macOS, Windows, and more.

Featured articles

[The Go Collective on Stack Overflow](#)

Published 23 June 2021

[Fuzzing is Beta Ready](#)

Published 3 June 2021

[Read more >](#)

Try Go

[Open in Playground ↗](#)

```
// You can edit this code!  
// Click here and start typing.  
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello, 世界")  
}
```

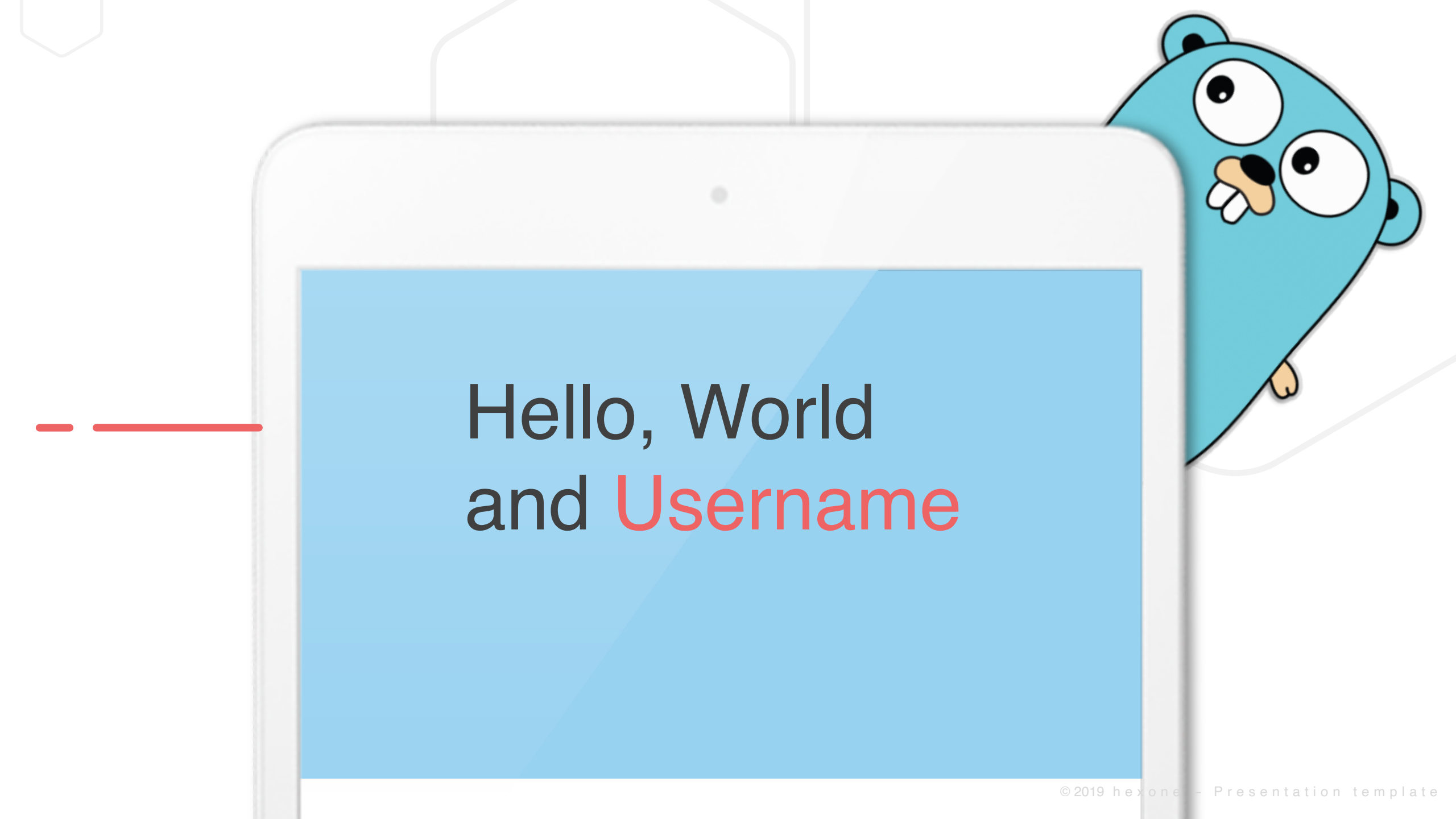
Hello, World! ▾

[Run](#)[Share](#)[Tour](#)

Featured video



ref: <https://golang.org>



Hello, World
and Username



Hello, World!

Create Directory call helloworld02:

1. mkdir helloworld02

Create file call main.go:

1. touch main.go
2. copy and paste the code in main.go

Run helloworld app:

1. go run main.go <your name>

Compile and Run helloworld app:

1. go build
2. ./helloworld02 <your name>

```
package main
```

```
import "fmt"
```

```
func main() {
```

```
    var name string = "World"
```

```
    if len(os.Args) > 1 {  
        name = os.Args[1]
```

```
    }
```

```
    fmt.Printf("Hello, %s!", name)
```

```
}
```



Primitive Types and Declarations



Declaration

Constant, Variables

```
package main

import "fmt"

func main() {
    x := 10
    fmt.Println(x)

    k, v := 1, 1
    fmt.Println(k, v)

    var k1, v1 int = 1, 1
    fmt.Println(k1, v1)

    var (
        k2 int = 1
        v2 string = "1"
    )
    fmt.Println(k2, v2)
}
```




Scope and Control Structure



Shadowing Variables

```
package main
```

```
import "fmt"
```

```
func main() {  
    x := 10  
    if x > 5 {  
        fmt.Println(x)  
        x := 5  
        fmt.Println(x)  
    }  
    fmt.Println(x)  
}
```



Shadowing Package

```
package main

import "fmt"

func main() {
    x := 10
    fmt.Println(x)
    fmt := "oops"
    fmt.Println(fmt)
}
```



Shadowing

Universe Block

```
package main

import "fmt"

func main() {
    fmt.Println(true)
    true := 10
    fmt.Println(true)

    string := false
    fmt.Println(string)
}
```



Unicode

Text, Emoji

```
package main
```

```
import "fmt"
```

```
func main() {
```

```
    var s string = "Hello 🐶"
```

```
    fmt.Println(len(s))
```

```
    fmt.Println(s[6:10])
```

```
    fmt.Println(len("🐶"))
```

```
}
```



Hello, World and Multi language



Hello, World!

Create Directory call helloworld03:

1. mkdir helloworld03

Create file call main.go:

1. touch main.go
2. copy and paste the code in main.go

Run helloworld app:

1. go run main.go

Compile and Run helloworld app:

1. go build
2. ./helloworld03

```
package main
```

```
import "fmt"
```

```
func main() {  
    if len(os.Args) < 2 {  
        os.Exit(1)  
    }  
    args := os.Args[1:]  
    greeting := "Hello"  
    if args[1] == "th" {  
        greeting = "สวัสดี"  
    }  
  
    fmt.Printf("%s, %s!\n", greeting, args[0])  
}
```



Composite Types



Slice Slicing

```
package main

import "fmt"

func main() {
    x := []int{1, 2, 3, 4}
    y := x[:2]
    z := x[1:]
    d := x[1:3]
    e := x[:]

    fmt.Println("x:", x)
    fmt.Println("y:", y)
    fmt.Println("z:", z)
    fmt.Println("d:", d)
    fmt.Println("e:", e)
}
```



this is our

creative **mockup**