免费电子书

学习

Go

Free unaffiliated eBook created from
Stack Overflow contributors.

#go

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: go

It is an unofficial and free Go ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Go.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Go

GoAlgolC。CSP。

。 。

1.8.3	2017524
1.8.0	2017216
1.7.0	2016815
1.6.0	2016217
1.5.0	2015819
1.4.0	2014124
1.3.0	2014618
1.2.0	2013121
1.1.0	2013513
1.0.0	2012-03-28

Examples

hello.go

```
package main

import "fmt"

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

Go

```
go run hello.go
```

```
Hello, 世界
```

```
go build hello.go
```

LinuxOSXUnix

```
./hello
```

```
hello.exe
```

Go。

FizzBuzz

“Hello World”FizzBuzz 。 FizzBuzz。 。

```
package main

// Simple fizzbuzz implementation

import "fmt"

func main() {
    for i := 1; i <= 100; i++ {
        s := ""
        if i % 3 == 0 {
            s += "Fizz"
        }
        if i % 5 == 0 {
            s += "Buzz"
        }
        if s != "" {
            fmt.Println(s)
        } else {
            fmt.Println(i)
        }
    }
}
```

Go

`go env [var ...]go`

```
$ go env
GOARCH="amd64"
GOBIN="/home/yourname/bin"
GOEXE=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/home/yourname"
GORACE=""
GOROOT="/usr/lib/go"
GOTOOLDIR="/usr/lib/go/pkg/tool/linux_amd64"
CC="gcc"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/tmp/go-
build059426571=/tmp/go-build -gno-record-gcc-switches"
CXX="g++"
CGO_ENABLED="1"
```

shell;。

```
$go env GOOS GOPATH
linux
/home/yourname
```

Gohttps://golang.org/dl/Go。

Gogo Goshell~/.profileUnix。

GOPATH

PATHGo path：；WindowsGo。 go get。

GOPATHGobin pkgsrc

- src - .go .c .g .s
- pkg - .a
- bin - Go

Go 1.8 GOPATH 。 Unix / Linux$ HOME / goWindowsUSERPROFILE/ go。

GOPATH。

GOBIN

go installgo getbinmain。 PATH。

GOROOT

Go。 。 Go。 GOROOT 。

```
godoc -http=:<port-number>
```

Go

```
go tool tour
```

Web。

fmt.Print

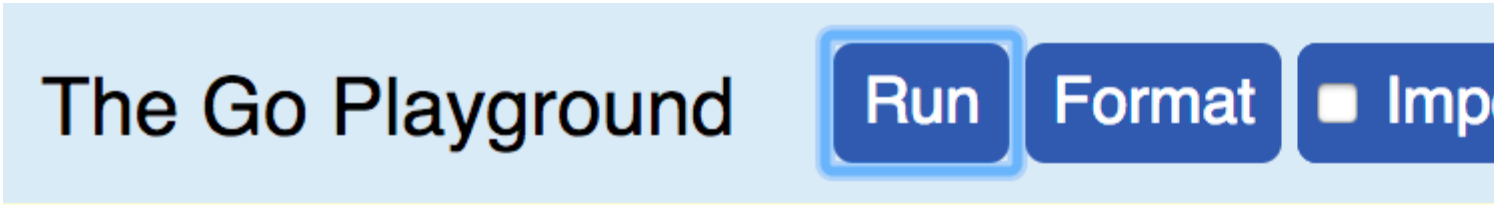
```
godoc cmd/fmt Print
# or
go doc fmt Print
```

```
go help [command]
```

The Go Playground 。Go。 。 。

1. Playground
2.
3. “”

Go Playground;“” 。




```
1  package main
2
3  import (
4          "fmt"
5  )
6
7  func main() {
8          fmt.Println("Hello, playgroun
9  }
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

2: Base64

- funcenc * base64.Encodingdstsrc [] byte
- funcenc * base64.EncodingDecodedstsrc [] byten interr error
- funcenc * base64.EncodingEncodeToStringsrc [] byte
- funcenc * base64.EncodingDecodeStrings string[] byteerror

encoding/base64 ◦ base64.StdEncoding base64.StdEncoding URLEncoding RawStdEncodign ◦

Examples

```
const foobar = `foo bar`
encoding := base64.StdEncoding
encodedFooBar := make([]byte, encoding.EncodedLen(len(foobar)))
encoding.Encode(encodedFooBar, []byte(foobar))
fmt.Printf("%s", encodedFooBar)
// Output: Zm9vIGJhcg==
```

```
str := base64.StdEncoding.EncodeToString([]byte(`foo bar`))
fmt.Println(str)
// Output: Zm9vIGJhcg==
```

```
encoding := base64.StdEncoding
data := []byte(`Zm9vIGJhcg==`)
decoded := make([]byte, encoding.DecodedLen(len(data)))
n, err := encoding.Decode(decoded, data)
if err != nil {
    log.Fatal(err)
}

// Because we don't know the length of the data that is encoded
// (only the max length), we need to trim the buffer to whatever
// the actual length of the decoded data was.
decoded = decoded[:n]

fmt.Printf("`%s`", decoded)
// Output: `foo bar`
```

```
decoded, err := base64.StdEncoding.DecodeString(`biws`)
if err != nil {
    log.Fatal(err)
}

fmt.Printf("%s", decoded)
// Output: n,,
```

Base64 https://riptutorial.com/zh-CN/go/topic/4492/base64

3: CGO

Examples

Cgo

Go C Bindings

GocgoC。 C APIC。

`import "C"` C

```
//#include <stdio.h>
import "C"
```

Go`stdio`。

C`"<>`

```
//#include "hello.c"
import "C"
```

```
    includeimport "C"
```

```
# command-line-arguments
could not determine kind of name for C.Hello
could not determine kind of name for C.sum
```

C。 C“”`hello.c`

```
//hello.c
#include <stdio.h>

void Hello(){
    printf("Hello world\n");
}
```

`sum.c`“hello world”

```
//sum.c
#include <stdio.h>

int sum(int a, int b) {
    return a + b;
}
```

...2。

`main.go`。

```
//main.go
package main

/*
  #include "hello.c"
  #include "sum.c"
*/
import "C"
```

GoC。Hello

```
//main.go
package main

/*
  #include "hello.c"
  #include "sum.c"
*/
import "C"


func main() {
    //Call to void function without params
    err := Hello()
    if err != nil {
        log.Fatal(err)
    }
}

//Hello is a C binding to the Hello World "C" program. As a Go user you could
//use now the Hello function transparently without knowing that it is calling
//a C function
func Hello() error {
    _, err := C.Hello()    //We ignore first result as it is a void function
    if err != nil {
        return errors.New("error calling Hello function: " + err.Error())
    }

    return nil
}
```

`go run main.go`main.goC“Hello world”。

。

```
//sum.c
#include <stdio.h>

int sum(int a, int b) {
  return a + b;
}
```

Go。

```
//main.go
package main

/*
#include "hello.c"
#include "sum.c"
*/
import "C"

import (
    "errors"
    "fmt"
    "log"
)

func main() {
    //Call to void function without params
    err := Hello()
    if err != nil {
        log.Fatal(err)
    }

    //Call to int function with two params
    res, err := makeSum(5, 4)
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("Sum of 5 + 4 is %d\n", res)
}

//Hello is a C binding to the Hello World "C" program. As a Go user you could
//use now the Hello function transparently without knowing that is calling a C
//function
func Hello() error {
    _, err := C.Hello() //We ignore first result as it is a void function
    if err != nil {
        return errors.New("error calling Hello function: " + err.Error())
    }

    return nil
}

//makeSum also is a C binding to make a sum. As before it returns a result and
//an error. Look that we had to pass the Int values to C.int values before using
//the function and cast the result back to a Go int value
func makeSum(a, b int) (int, error) {
    //Convert Go ints to C ints
    aC := C.int(a)
    bC := C.int(b)

    sum, err := C.sum(aC, bC)
    if err != nil {
        return 0, errors.New("error calling Sum function: " + err.Error())
    }

    //Convert C.int result to Go int
    res := int(sum)

    return res, nil
}
```

“makeSum”。 `C.int`C `int int`。 C `int`。 `int()`GoGoint。

`go run main.go`go

```
$ go run main.go
Hello world!
Sum of 5 + 4 is 9
```

---

go build。

```
$ go build
# github.com/sayden/c-bindings
/tmp/go-build329491076/github.com/sayden/c-bindings/_obj/hello.o: In function `Hello':
../../go/src/github.com/sayden/c-bindings/hello.c:5: multiple definition of `Hello'
/tmp/go-build329491076/github.com/sayden/c-
bindings/_obj/main.cgo2.o:/home/mariocaster/go/src/github.com/sayden/c-bindings/hello.c:5:
first defined here
/tmp/go-build329491076/github.com/sayden/c-bindings/_obj/sum.o: In function `sum':
../../go/src/github.com/sayden/c-bindings/sum.c:5: multiple definition of `sum`
/tmp/go-build329491076/github.com/sayden/c-
bindings/_obj/main.cgo2.o:/home/mariocaster/go/src/github.com/sayden/c-bindings/sum.c:5: first
defined here
collect2: error: ld returned 1 exit status
```

`go build`

```
$ go build main.go
$ ./main
Hello world!
Sum of 5 + 4 is 9
```

`-o` flag `go build -o my_c_binding main.go`

。

CGO https://riptutorial.com/zh-CN/go/topic/6125/cgo

4: CGO

Examples

GoC

CgoCGo。
cgo“C”Go。 GoC.intC.Add。
“C”C。
cgoimport。
import "C"“”。 import

```
import "C"
import "fmt"
```

factored import

```
import "C"
import (
    "fmt"
    "math"
)
```

cgo

```
package main

//int Add(int a, int b){
//    return a+b;
//}
import "C"
import "fmt"

func main() {
    a := C.int(10)
    b := C.int(20)
    c := C.Add(a, b)
    fmt.Println(c) // 30
}
```

go build

```
30
```

cgogo buildgo install。 go tool"C"cgo。

CGo

GoC

```
package main

/*
// Everything in comments above the import "C" is C code and will be compiles with the GCC.
// Make sure you have a GCC installed.

int addInC(int a, int b) {
    return a + b;
}
 */
import "C"
import "fmt"

func main() {
       a := 3
       b := 5

       c := C.addInC(C.int(a), C.int(b))

       fmt.Println("Add in C:", a, "+", b, "=", int(c))
}
```

CGo

```
package main

/*
static inline int multiplyInGo(int a, int b) {
    return go_multiply(a, b);
}
 */
import "C"
import (
       "fmt"
)

func main() {
       a := 3
       b := 5

       c := C.multiplyInGo(C.int(a), C.int(b))

       fmt.Println("multiplyInGo:", a, "*", b, "=", int(c))
}

//export go_multiply
func go_multiply(a C.int, b C.int) C.int {
       return a * b
}
```

```
package main

/*
int go_multiply(int a, int b);

typedef int (*multiply_f)(int a, int b);
multiply_f multiply;

static inline init() {
```

```
    multiply = go_multiply;
}

static inline int multiplyWithFp(int a, int b) {
    return multiply(a, b);
}
 */
import "C"
import (
       "fmt"
)

func main() {
       a := 3
       b := 5
       C.init(); // OR:
       C.multiply = C.multiply_f(go_multiply);

       c := C.multiplyWithFp(C.int(a), C.int(b))

       fmt.Println("multiplyInGo:", a, "+", b, "=", int(c))
}

//export go_multiply
func go_multiply(a C.int, b C.int) C.int {
       return a * b
}
```

Go

```
// Go string to C string
// The C string is allocated in the C heap using malloc.
// It is the caller's responsibility to arrange for it to be
// freed, such as by calling C.free (be sure to include stdlib.h
// if C.free is needed).
func C.CString(string) *C.char

// Go []byte slice to C array
// The C array is allocated in the C heap using malloc.
// It is the caller's responsibility to arrange for it to be
// freed, such as by calling C.free (be sure to include stdlib.h
// if C.free is needed).
func C.CBytes([]byte) unsafe.Pointer

// C string to Go string
func C.GoString(*C.char) string

// C data with explicit length to Go string
func C.GoStringN(*C.char, C.int) string

// C data with explicit length to Go []byte
func C.GoBytes(unsafe.Pointer, C.int) []byte
```

```
func go_handleData(data *C.uint8_t, length C.uint8_t) []byte {
       return C.GoBytes(unsafe.Pointer(data), C.int(length))
}

// ...
```

```
goByteSlice := []byte {1, 2, 3}
goUnsafePointer := C.CBytes(goByteSlice)
cPointer := (*C.uint8_t)(goUnsafePointer)

// ...

func getPayload(packet *C.packet_t) []byte {
       dataPtr := unsafe.Pointer(packet.data)
       // Lets assume a 2 byte header before the payload.
       payload := C.GoBytes(unsafe.Pointer(uintptr(dataPtr)+2), C.int(packet.dataLength-2))
       return payload
}
```

CGO https://riptutorial.com/zh-CN/go/topic/6455/cgo

5: FMT

Examples

`fmt.Stringer` `String()` `string`。 string“”`fmt`。

```
package main

import (
    "fmt"
)

type User struct {
    Name  string
    Email string
}

// String satisfies the fmt.Stringer interface for the User type
func (u User) String() string {
    return fmt.Sprintf("%s <%s>", u.Name, u.Email)
}

func main() {
    u := User{
        Name:  "John Doe",
        Email: "johndoe@example.com",
    }

    fmt.Println(u)
    // output: John Doe <johndoe@example.com>
}
```

Playground

fmt

fmtI / O

```
%v    // the value in a default format
%T    // a Go-syntax representation of the type of the value
%s    // the uninterpreted bytes of the string or slice
```

`fmt`**4**。

```
fmt.Print("Hello World")        // prints: Hello World
fmt.Println("Hello World")      // prints: Hello World\n
fmt.Printf("Hello %s", "World") // prints: Hello World
```

```
formattedString := fmt.Sprintf("%v %s", 2, "words") // returns string "2 words"
```

Fprint

```
byteCount, err := fmt.Fprint(w, "Hello World") // writes to io.Writer w
```

`httpFprint`

```
func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello %s!", "Browser")
}   // Writes: "Hello Browser!" onto http response
```

。

```
var s string
fmt.Scanln(&s) // pass pointer to buffer
// Scanln is similar to fmt.Scan(), but it stops scanning at new line.
fmt.Println(s) // whatever was inputted
```

Stringer

`String()` `fmt` **inteface** `Stringer`

```
type Stringer interface {
        String() string
}
```

FMT https://riptutorial.com/zh-CN/go/topic/2938/fmt

6: GoJWT

JSON WebJWT。Web。

context.ContextHTTPhttps://github.com/goware/jwtauth https://github.com/auth0/go-jwt- https://github.com/dgrijalva/jwt-go 。

Dave Grijalvago-jwt。

Examples

HMAC

```
// sample token string taken from the New example
tokenString :=
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmb28iOiJiYXIiLCJuYmYiOjE0NDQ0Nzg0MDB9.u1riaD1rW97opCoAuRCTy4w5
Zk-bh7vLiRIsrpU"

// Parse takes the token string and a function for looking up the key. The latter is
especially
// useful if you use multiple keys for your application.  The standard is to use 'kid' in the
// head of the token to identify which key to use, but the parsed token (head and claims) is
provided
// to the callback, providing flexibility.
token, err := jwt.Parse(tokenString, func(token *jwt.Token) (interface{}, error) {
    // Don't forget to validate the alg is what you expect:
    if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
        return nil, fmt.Errorf("Unexpected signing method: %v", token.Header["alg"])
    }

    // hmacSampleSecret is a []byte containing your secret, e.g. []byte("my_secret_key")
    return hmacSampleSecret, nil
})

if claims, ok := token.Claims.(jwt.MapClaims); ok && token.Valid {
    fmt.Println(claims["foo"], claims["nbf"])
} else {
    fmt.Println(err)
}
```

```
bar 1.4444784e+09
```

Dave Grijalva。

`StandardClaim`。

```
tokenString :=
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmb28iOiJiYXIiLCJleHAiOjE1MDAwLCJpc3MiOiJ0ZXN0In0.HE7fK0xOQwFEr


type MyCustomClaims struct {
```

```
    Foo string `json:"foo"`
    jwt.StandardClaims
}

// sample token is expired.  override time so it parses as valid
at(time.Unix(0, 0), func() {
    token, err := jwt.ParseWithClaims(tokenString, &MyCustomClaims{}, func(token *jwt.Token)
(interface{}, error) {
        return []byte("AllYourBase"), nil
    })

    if claims, ok := token.Claims.(*MyCustomClaims); ok && token.Valid {
        fmt.Printf("%v %v", claims.Foo, claims.StandardClaims.ExpiresAt)
    } else {
        fmt.Println(err)
    }
})
```

```
bar 15000
```

Dave Grijalva。

HMACJWT

```
// Create a new token object, specifying signing method and the claims
// you would like it to contain.
token := jwt.NewWithClaims(jwt.SigningMethodHS256, jwt.MapClaims{
    "foo": "bar",
    "nbf": time.Date(2015, 10, 10, 12, 0, 0, 0, time.UTC).Unix(),
})

// Sign and get the complete encoded token as a string using the secret
tokenString, err := token.SignedString(hmacSampleSecret)

fmt.Println(tokenString, err)
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmb28iOiJiYXIiLCJuYmYiOjE0NDQ0Nzg0MDB9.u1riaD1rW97opCoAuRCTy4w58
Zk-bh7vLiRIsrpU <nil>
```

Dave Grijalva。

StandardClaims

StandardClaims。 。 。

```
mySigningKey := []byte("AllYourBase")

// Create the Claims
claims := &jwt.StandardClaims{
    ExpiresAt: 15000,
    Issuer:    "test",
}

token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
```

```
ss, err := token.SignedString(mySigningKey)
fmt.Printf("%v %v", ss, err)
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1MDAwLCJpc3MiOiJ0ZXN0In0.QsODzZu3lUZMVdhbO76u3Jv02iYCvE
<nil>
```

Dave Grijalva。

```
// Token from another example.  This token is expired
var tokenString =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmb28iOiJiYXIiLCJleHAiOjE1MDAwLCJpc3MiOiJ0ZXN0In0.HE7fK0xOQwFEr


token, err := jwt.Parse(tokenString, func(token *jwt.Token) (interface{}, error) {
    return []byte("AllYourBase"), nil
})

if token.Valid {
    fmt.Println("You look nice today")
} else if ve, ok := err.(*jwt.ValidationError); ok {
    if ve.Errors&jwt.ValidationErrorMalformed != 0 {
        fmt.Println("That's not even a token")
    } else if ve.Errors&(jwt.ValidationErrorExpired|jwt.ValidationErrorNotValidYet) != 0 {
        // Token is either expired or not active yet
        fmt.Println("Timing is everything")
    } else {
        fmt.Println("Couldn't handle this token:", err)
    }
} else {
    fmt.Println("Couldn't handle this token:", err)
}
```

```
Timing is everything
```

Dave Grijalva。

HTTP Authorization

```
type contextKey string

const (
    // JWTTokenContextKey holds the key used to store a JWT Token in the
    // context.
    JWTTokenContextKey contextKey = "JWTToken"

    // JWTClaimsContextKey holds the key used to store the JWT Claims in the
    // context.
    JWTClaimsContextKey contextKey = "JWTClaims"
)

// ToHTTPContext moves JWT token from request header to context.
func ToHTTPContext() http.RequestFunc {
    return func(ctx context.Context, r *stdhttp.Request) context.Context {
        token, ok := extractTokenFromAuthHeader(r.Header.Get("Authorization"))
        if !ok {
```

```
            return ctx
        }

        return context.WithValue(ctx, JWTTokenContextKey, token)
    }
}
```

go-kit / kit Peter Bourgon

GoJWT https://riptutorial.com/zh-CN/go/topic/10161/gojwt

7: GoProtobuf

ProtobufProtocol Buffer。 protobuf。 。

protobuf。

1.
2. 。

gRPC

protoRPCprotoc-gen-gogRPC http://www.grpc.io/ 。 `plugins`protoc-gen-go;protoc--go_out

```
protoc --go_out=plugins=grpc:. *.proto
```

Examples

ProtobufGo

test.proto

```
package example;

enum FOO { X = 17; };

message Test {
  required string label = 1;
  optional int32 type = 2 [default=77];
  repeated int64 reps = 3;
  optional group OptionalGroup = 4 {
    required string RequiredField = 5;
  }
}
```

protoc--go_outGo。

```
protoc --go_out=. *.proto
```

Test

```
package main

import (
    "log"

    "github.com/golang/protobuf/proto"
    "path/to/example"
)

func main() {
```

```
    test := &example.Test {
        Label: proto.String("hello"),
        Type:  proto.Int32(17),
        Reps:  []int64{1, 2, 3},
        Optionalgroup: &example.Test_OptionalGroup {
            RequiredField: proto.String("good bye"),
        },
    }
    data, err := proto.Marshal(test)
    if err != nil {
        log.Fatal("marshaling error: ", err)
    }
    newTest := &example.Test{}
    err = proto.Unmarshal(data, newTest)
    if err != nil {
        log.Fatal("unmarshaling error: ", err)
    }
    // Now test and newTest contain the same data.
    if test.GetLabel() != newTest.GetLabel() {
        log.Fatalf("data mismatch %q != %q", test.GetLabel(), newTest.GetLabel())
    }
    // etc.
}
```

```
protoc --go_out=plugins=grpc,import_path=mypackage:. *.proto
```

GoProtobuf https://riptutorial.com/zh-CN/go/topic/9729/goprotobuf

8: Go

goGo。 Go。

Examples

go run。 。 run **main** 。

Hello Worldmain.go

```
package main

import fmt

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

```
go run main.go
```

```
Hello, World!
```

mainrun

```
go run main.go assets.go
```

go build。

Hello Worldmain.go

```
package main

import fmt

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

```
go build main.go
```

build mainmain.exe。 Hello, World! 。 Go 。

OSArchitecture

buildenv

```
env GOOS=linux go build main.go # builds for Linux
```

```
env GOARCH=arm go build main.go # builds for ARM architecture
```

main

```
go build main.go assets.go # outputs an executable: main
```

`main`

```
go build . # outputs an executable with name as the name of enclosing folder
```

`go clean` `go build` Makefile

Fmt

`go fmt` `go fmt`

```
go fmt main.go
```

```
go fmt myProject
```

`gofmt -s` `go fmt`

`gofmt` `go fmt` Go `main.go`

```
package main

type Example struct {
    Name string
}

func (e *Example) Original(name string) {
    e.Name = name
}

func main() {
    e := &Example{"Hello"}
    e.Original("Goodbye")
}
```

`gofmt` `Original` with `Refactor`

```
gofmt -r 'Original -> Refactor' -d main.go
```

```
-func (e *Example) Original(name string) {
+func (e *Example) Refactor(name string) {
     e.Name = name
 }

 func main() {
     e := &Example{"Hello"}
```

```
-    e.Original("Goodbye")
+    e.Refactor("Goodbye")
 }
```

go get。 'go install'。 Get。

github.com/maknahar/phonecountry

get$GOPATH/src/<import-path> 。 GOPATHget。 $GOPATH/bin。

Go。 “go1”“go1”。 。

go get -d。 -u。

。

go env [var ...]。

。

$go env

```
GOARCH="amd64"
GOBIN=""
GOEXE=""
GOHOSTARCH="amd64"
GOHOSTOS="darwin"
GOOS="darwin"
GOPATH="/Users/vikashkv/work"
GORACE=""
GOROOT="/usr/local/Cellar/go/1.7.4_1/libexec"
GOTOOLDIR="/usr/local/Cellar/go/1.7.4_1/libexec/pkg/tool/darwin_amd64"
CC="clang"
GOGCCFLAGS="-fPIC -m64 -pthread -fno-caret-diagnostics -Qunused-arguments -fmessage-length=0 -
fdebug-prefix-map=/var/folders/xf/t3j24fjd2b7bv8c9gdr_0mj80000gn/T/go-build785167995=/tmp/go-
build -gno-record-gcc-switches -fno-common"
CXX="clang++"
CGO_ENABLED="1"
```

。

$go env GOOS GOPATH

```
darwin
/Users/vikashkv/work
```

Go https://riptutorial.com/zh-CN/go/topic/4828/go

9: HTTP

- resperr= http.Geturl//HTTPHTTP GET。。
- resperr= http.PosturlbodyTypebody//HTTPHTTP POST。。
- resperr= http.PostFormurlvalues//HTTPHTTPPOST。。

RESP	*http.ResponseHTTP
	error。 nil。
	HTTPstringURL。
	POSTstringMIME。
	io.Reader Read() POST。
	url.Valuesurl.Values。 map[string][]string 。

HTTPdefer resp.Body.Close()。

Examples

GET

GETHTML。

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
)

func main() {
    resp, err := http.Get("https://example.com/")
    if err != nil {
        panic(err)
    }

    // It is important to defer resp.Body.Close(), else resource leaks will occur.
    defer resp.Body.Close()

    data, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        panic(err)
    }

    // Will print site contents (HTML) to output
```

```
    fmt.Println(string(data))
}
```

URLJSON

Stack Exchange API10StackOverflow。

```
package main

import (
    "encoding/json"
    "fmt"
    "net/http"
    "net/url"
)

const apiURL = "https://api.stackexchange.com/2.2/posts?"

// Structs for JSON decoding
type postItem struct {
    Score int    `json:"score"`
    Link  string `json:"link"`
}

type postsType struct {
    Items []postItem `json:"items"`
}

func main() {
    // Set URL parameters on declaration
    values := url.Values{
        "order": []string{"desc"},
        "sort":  []string{"activity"},
        "site":  []string{"stackoverflow"},
    }

    // URL parameters can also be programmatically set
    values.Set("page", "1")
    values.Set("pagesize", "10")

    resp, err := http.Get(apiURL + values.Encode())
    if err != nil {
        panic(err)
    }

    defer resp.Body.Close()

    // To compare status codes, you should always use the status constants
    // provided by the http package.
    if resp.StatusCode != http.StatusOK {
        panic("Request was not OK: " + resp.Status)
    }

    // Example of JSON decoding on a reader.
    dec := json.NewDecoder(resp.Body)
    var p postsType
    err = dec.Decode(&p)
    if err != nil {
        panic(err)
```

```
    }

    fmt.Println("Top 10 most recently active StackOverflow posts:")
    fmt.Println("Score", "Link")
    for _, post := range p.Items {
        fmt.Println(post.Score, post.Link)
    }
}
```

1.7+

HTTP1.7+

```
import (
    "context"
    "net/http"
    "time"
)

req, err := http.NewRequest("GET", `https://example.net`, nil)
ctx, _ := context.WithTimeout(context.TODO(), 200 * time.Milliseconds)
resp, err := http.DefaultClient.Do(req.WithContext(ctx))
// Be sure to handle errors.
defer resp.Body.Close()
```

1.7

```
import (
    "net/http"
    "time"

    "golang.org/x/net/context"
    "golang.org/x/net/context/ctxhttp"
)

ctx, err := context.WithTimeout(context.TODO(), 200 * time.Milliseconds)
resp, err := ctxhttp.Get(ctx, http.DefaultClient, "https://www.example.net")
// Be sure to handle errors.
defer resp.Body.Close()
```

context ◦

PUTJSON

PUTUser

```
package main

import (
    "bytes"
    "encoding/json"
    "fmt"
    "net/http"
```

```
)

type User struct {
    Name  string
    Email string
}

func main() {
    user := User{
        Name:  "John Doe",
        Email: "johndoe@example.com",
    }

    // initialize http client
    client := &http.Client{}

    // marshal User to json
    json, err := json.Marshal(user)
    if err != nil {
        panic(err)
    }

    // set the HTTP method, url, and request body
    req, err := http.NewRequest(http.MethodPut, "http://api.example.com/v1/user",
bytes.NewBuffer(json))
    if err != nil {
        panic(err)
    }

    // set the request header Content-Type for json
    req.Header.Set("Content-Type", "application/json; charset=utf-8")
    resp, err := client.Do(req)
    if err != nil {
        panic(err)
    }

    fmt.Println(resp.StatusCode)
}
```

HTTP https://riptutorial.com/zh-CN/go/topic/1422/http

10: HTTP

`http.ServeMux`HTTP。

- Mux

Examples

HTTP Hello World

```
package main

import (
    "log"
    "net/http"
)

func main() {

    // Create a mux for routing incoming requests
    m := http.NewServeMux()

    // All URLs will be handled by this function
    m.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Hello, world!"))
    })

    // Create a server listening on port 8000
    s := &http.Server{
        Addr:    ":8000",
        Handler: m,
    }

    // Continue to process new requests until an error occurs
    log.Fatal(s.ListenAndServe())
}
```

`Ctrl` + `C`。

golangweb`net/http`。

。

。 HTTP。 HTTP`"Hello World"`。

`server.go`。

```
package main

import (
    "log"
    "net/http"
```

```
)

func main() {
    // All URLs will be handled by this function
    // http.HandleFunc uses the DefaultServeMux
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Hello, world!"))
    })

    // Continue to process new requests until an error occurs
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

```
$ go run server.go
```

。

```
$ go build server.go
$ ./server
```

:8080 。 HTTP。 cURL

```
curl -i http://localhost:8080/
HTTP/1.1 200 OK
Date: Wed, 20 Jul 2016 18:04:46 GMT
Content-Length: 13
Content-Type: text/plain; charset=utf-8

Hello, world!
```

Ctrl + C。

HandleFuncmux。

Hello World

```
http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintln(w, "Hello, world!")
}
```

HandlerFunc。

```
func FunctionName(w http.ResponseWriter, req *http.Request)
```

HandlerFunc 。

```
package main

import (
    "fmt"
    "net/http"
)
```

```
// A HandlerFunc function
// Notice the signature of the function
func RootHandler(w http.ResponseWriter, req *http.Request) {
    fmt.Fprintln(w, "Hello, world!")
}

func main() {
    // Here we pass the reference to the `RootHandler` handler function
    http.HandleFunc("/", RootHandler)
    panic(http.ListenAndServe(":8080", nil))
}
```

。

```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func FooHandler(w http.ResponseWriter, req *http.Request) {
    fmt.Fprintln(w, "Hello from foo!")
}

func BarHandler(w http.ResponseWriter, req *http.Request) {
    fmt.Fprintln(w, "Hello from bar!")
}

func main() {
    http.HandleFunc("/foo", FooHandler)
    http.HandleFunc("/bar", BarHandler)

    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

cURL

```
➜  ~ curl -i localhost:8080/foo
HTTP/1.1 200 OK
Date: Wed, 20 Jul 2016 18:23:08 GMT
Content-Length: 16
Content-Type: text/plain; charset=utf-8

Hello from foo!

➜  ~ curl -i localhost:8080/bar
HTTP/1.1 200 OK
Date: Wed, 20 Jul 2016 18:23:10 GMT
Content-Length: 16
Content-Type: text/plain; charset=utf-8

Hello from bar!

➜  ~ curl -i localhost:8080/
HTTP/1.1 404 Not Found
Content-Type: text/plain; charset=utf-8
```

```
X-Content-Type-Options: nosniff
Date: Wed, 20 Jul 2016 18:23:13 GMT
Content-Length: 19

404 page not found
```

HTTPS

HTTPS。 `openssl`

```
openssl req -x509 -newkey rsa:4096 -sha256 -nodes -keyout key.pem -out cert.pem -subj
"/CN=example.com" -days 3650`
```

- `req`
- `x509`
- `newkey rsa:40964096`RSA
- `sha256`SHA2562017
- `nodes`。 。
- `keyout`
- `out`
- `subj`
- `days` Fow `3650`。 10。

。 。 。 。 “Let's Encrypt” https //letsencrypt.org

Go

TLS。 `cert.pem``key.pem`SSL。

```
package main

import (
    "log"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Hello, world!"))
    })

    log.Fatal(http.ListenAndServeTLS(":443","cert.pem","key.pem", nil))
}
```

HTTP

Go`http.ResponseWriter`。 。

GoGo。

`html/template`HTTP

```
package main

import(
    "html/template"
    "net/http"
    "log"
)

func main(){
    http.HandleFunc("/",WelcomeHandler)
    http.ListenAndServe(":8080",nil)
}

type User struct{
    Name string
    nationality string //unexported field.
}

func check(err error){
    if err != nil{
        log.Fatal(err)
    }
}

func WelcomeHandler(w http.ResponseWriter, r *http.Request){
    if r.Method == "GET"{
        t,err := template.ParseFiles("welcomeform.html")
        check(err)
        t.Execute(w,nil)
    }else{
        r.ParseForm()
        myUser := User{}
        myUser.Name = r.Form.Get("entered_name")
        myUser.nationality = r.Form.Get("entered_nationality")
        t, err := template.ParseFiles("welcomeresponse.html")
        check(err)
        t.Execute(w,myUser)
    }
}
```

1. `welcomeform.html`

```
<head>
    <title> Help us greet you </title>
</head>
<body>
    <form method="POST" action="/">
        Enter Name: <input type="text" name="entered_name">
        Enter Nationality: <input type="text" name="entered_nationality">
        <input type="submit" value="Greet me!">
    </form>
</body>
```

1. `welcomeresponse.html`

```
<head>
```

```
    <title> Greetings, {{.Name}} </title>
</head>
<body>
    Greetings, {{.Name}}.<br>
    We know you are a {{.nationality}}!
</body>
```

1. .html。
2. http://localhost:8080/ 。
3. 。

ServeMux

```
package main

import (
    "net/http"
)

func main() {
    muxer := http.NewServeMux()
    fileServerCss := http.FileServer(http.Dir("src/css"))
    fileServerJs := http.FileServer(http.Dir("src/js"))
    fileServerHtml := http.FileServer(http.Dir("content"))
    muxer.Handle("/", fileServerHtml)
    muxer.Handle("/css", fileServerCss)
    muxer.Handle("/js", fileServerJs)
    http.ListenAndServe(":8080", muxer)
}
```

http

APIHTTP。

- http.Handler
- http.ResponseWriter
- http.Request
-

```
package main

import (
    "fmt"
    "io/ioutil"
    "log"
    "net/http"
)

type customHandler struct{}

// ServeHTTP implements the http.Handler interface in the net/http package
func (h customHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
```

```
    // ParseForm will parse query string values and make r.Form available
    r.ParseForm()

    // r.Form is map of query string parameters
    // its' type is url.Values, which in turn is a map[string][]string
    queryMap := r.Form

    switch r.Method {
    case http.MethodGet:
        // Handle GET requests
        w.WriteHeader(http.StatusOK)
        w.Write([]byte(fmt.Sprintf("Query string values: %s", queryMap)))
        return
    case http.MethodPost:
        // Handle POST requests
        body, err := ioutil.ReadAll(r.Body)
        if err != nil {
            // Error occurred while parsing request body
            w.WriteHeader(http.StatusBadRequest)
            return
        }
        w.WriteHeader(http.StatusOK)
        w.Write([]byte(fmt.Sprintf("Query string values: %s\nBody posted: %s", queryMap,
body)))
        return
    }

    // Other HTTP methods (eg PUT, PATCH, etc) are not handled by the above
    // so inform the client with appropriate status code
    w.WriteHeader(http.StatusMethodNotAllowed)
}

func main() {
    // All URLs will be handled by this function
    // http.Handle, similarly to http.HandleFunc
    // uses the DefaultServeMux
    http.Handle("/", customHandler{})

    // Continue to process new requests until an error occurs
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

```
$ curl -i 'localhost:8080?city=Seattle&state=WA' -H 'Content-Type: text/plain' -X GET
HTTP/1.1 200 OK
Date: Fri, 02 Sep 2016 16:36:24 GMT
Content-Length: 51
Content-Type: text/plain; charset=utf-8

Query string values: map[city:[Seattle] state:[WA]]%

$ curl -i 'localhost:8080?city=Seattle&state=WA' -H 'Content-Type: text/plain' -X POST -d
"some post data"
HTTP/1.1 200 OK
Date: Fri, 02 Sep 2016 16:36:35 GMT
Content-Length: 79
Content-Type: text/plain; charset=utf-8

Query string values: map[city:[Seattle] state:[WA]]
Body posted: some post data%
```

```
$ curl -i 'localhost:8080?city=Seattle&state=WA' -H 'Content-Type: text/plain' -X PUT
HTTP/1.1 405 Method Not Allowed
Date: Fri, 02 Sep 2016 16:36:41 GMT
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

HTTP https://riptutorial.com/zh-CN/go/topic/756/http

11: IOTA

Iota。 Iota。

`iota`。 iota1`const`。 iota `iota + 1` 。

Examples

iota

- `iota`

```
const (
  a = iota // a = 0
  b = iota // b = 1
  c = iota // c = 2
)
```

- `iota`

```
const (
  a = iota // a = 0
  b        // b = 1
  c        // c = 2
)
```

iota

`iota`。 SIEffective Go

```
type ByteSize float64

const (
    _           = iota // ignore first value by assigning to blank identifier
    KB ByteSize = 1 << (10 * iota)
    MB
    GB
    TB
    PB
    EB
    ZB
    YB
)
```

iota `iota`

```
const ( // iota is reset to 0
    a = 1 << iota  // a == 1
    b = 1 << iota  // b == 2
    c = 3          // c == 3  (iota is not used but still incremented)
```

```
    d = 1 << iota  // d == 8
)
```

```
const (
  a = iota // a = 0
  _        // iota is incremented
  b        // b = 2
)
```

Go Spec CC-BY 3.0。

iota

iotaConstSpecConstSpec iota

```
const (
    bit0, mask0 = 1 << iota, 1<<iota - 1  // bit0 == 1, mask0 == 0
    bit1, mask1                           // bit1 == 2, mask1 == 1
    _, _                                  // skips iota == 2
    bit3, mask3                           // bit3 == 8, mask3 == 7
)
```

Go Spec CC-BY 3.0。

iota

Iota。 /

```
const (
    Secure = 1 << iota // 0b001
    Authn              // 0b010
    Ready              // 0b100
)

ConnState := Secure|Authn // 0b011: Connection is secure and authenticated, but not yet Ready
```

constiota

const。 Go0iota1。 。 iota。

constiota

```
package main

import "fmt"

const (
    Low = 5 * iota
    Medium
    High
)
```

```
func main() {
    // Use our iota constants.
    fmt.Println(Low)
    fmt.Println(Medium)
    fmt.Println(High)
}
```

Go Playground

IOTA https://riptutorial.com/zh-CN/go/topic/2865/iota

12: JSON

- func Marshalv interface {}[] byteerror
- func Unmarshaldata [] bytev interface {}

"encoding/json"jsonGoJSON。

JSONGo

JSON	
	float64int

Examples

JSON

"encoding/json"json.MarshalJSON。

。JSON。

```
decodedValue := []string{"foo", "bar"}

// encode the value
data, err := json.Marshal(decodedValue)

// check if the encoding is successful
if err != nil {
    panic(err)
}

// print out the JSON-encoded string
// remember that data is a []byte
fmt.Println(string(data))
// "["foo","bar"]"
```

```
var data []byte

data, _ = json.Marshal(1)
fmt.Println(string(data))
// 1

data, _ = json.Marshal("1")
fmt.Println(string(data))
// "1"

data, _ = json.Marshal(true)
```

```
fmt.Println(string(data))
// true

data, _ = json.Marshal(map[string]int{"London": 18, "Rome": 30})
fmt.Println(string(data))
// {"London":18,"Rome":30}
```

JSONGo。 。

JSON

"encoding/json"json.UnmarshalJSON。

[]bytes。 。

```
encodedValue := []byte(`{"London":18,"Rome":30}`)

// generic storage for the decoded JSON
var data map[string]interface{}

// decode the value into data
// notice that we must pass the pointer to data using &data
err := json.Unmarshal(encodedValue, &data)

// check if the decoding is successful
if err != nil {
    panic(err)
}

fmt.Println(data)
map[London:18 Rome:30]
```

。 。 JSON。

```
encodedValue := []byte(`{"city":"Rome","temperature":30}`)

// generic storage for the decoded JSON
var data map[string]interface{}

// decode the value into data
if err := json.Unmarshal(encodedValue, &data); err != nil {
    panic(err)
}

// if you want to use a specific value type, we need to cast it
temp := data["temperature"].(float64)
fmt.Println(temp) // 30
city := data["city"].(string)
fmt.Println(city) // "Rome"
```

。 map[string]interface{}。

。 JSONstruct 。

JSON

JSON。

data.json

```
[
    {
      "Name" : "John Doe",
      "Standard" : 4
    },
    {
      "Name" : "Peter Parker",
      "Standard" : 11
    },
    {
      "Name" : "Bilbo Baggins",
      "Standard" : 150
    }
]
```

```
package main

import (
    "encoding/json"
    "fmt"
    "log"
    "os"
)

type Student struct {
    Name     string
    Standard int `json:"Standard"`
}

func main() {
    // open the file pointer
    studentFile, err := os.Open("data.json")
    if err != nil {
        log.Fatal(err)
    }
    defer studentFile.Close()

    // create a new decoder
    var studentDecoder *json.Decoder = json.NewDecoder(studentFile)
    if err != nil {
        log.Fatal(err)
    }

    // initialize the storage for the decoded data
    var studentList []Student

    // decode the data
    err = studentDecoder.Decode(&studentList)
    if err != nil {
        log.Fatal(err)
    }

    for i, student := range studentList {
        fmt.Println("Student", i+1)
        fmt.Println("Student name:", student.Name)
        fmt.Println("Student standard:", student.Standard)
```

```
    }
}
```

`data.json`Go。 GoGo File I / O 。

。

```
jsonBlob := []byte(`
  {
    "_total": 1,
    "_links": {
      "self":
"https://api.twitch.tv/kraken/channels/foo/subscriptions?direction=ASC&limit=25&offset=0",
      "next":
"https://api.twitch.tv/kraken/channels/foo/subscriptions?direction=ASC&limit=25&offset=25"
    },
    "subscriptions": [
      {
        "created_at": "2011-11-23T02:53:17Z",
        "_id": "abcdef0000000000000000000000000000000000",
        "_links": {
          "self": "https://api.twitch.tv/kraken/channels/foo/subscriptions/bar"
        },
        "user": {
          "display_name": "bar",
          "_id": 123456,
          "name": "bar",
          "staff": false,
          "created_at": "2011-06-16T18:23:11Z",
          "updated_at": "2014-10-23T02:20:51Z",
          "logo": null,
          "_links": {
            "self": "https://api.twitch.tv/kraken/users/bar"
          }
        }
      }
    ]
  }
`)

var js struct {
    Total int `json:"_total"`
    Links struct {
        Next string `json:"next"`
    } `json:"_links"`
    Subs []struct {
        Created string `json:"created_at"`
        User    struct {
            Name string `json:"name"`
            ID   int    `json:"_id"`
        } `json:"user"`
    } `json:"subscriptions"`
}

err := json.Unmarshal(jsonBlob, &js)
if err != nil {
    fmt.Println("error:", err)
}
fmt.Printf("%+v", js)
```

```
{Total:1
Links:{Next:https://api.twitch.tv/kraken/channels/foo/subscriptions?direction=ASC&limit=25&offset=25}
Subs:[{Created:2011-11-23T02:53:17Z User:{Name:bar ID:123456}}]}
```

http //stackoverflow.com/documentation/go/994/json/4111/encoding-decoding-go-structs

JSON

```
type Company struct {
    Name     string
    Location string
}
```

/

`Revenue` and `Sales` /`json:"-"`。 。

```
type Company struct {
    Name     string `json:"name"`
    Location string `json:"location"`
    Revenue  int    `json:"-"`
    sales    int
}
```

`Location`JSONJSON,`omitempty`json,`omitempty` 。

```
type Company struct {
    Name     string `json:"name"`
    Location string `json:"location,omitempty"`
}
```

```
type MyStruct struct {
    uuid string
    Name string
}
```

https //play.golang.org/p/Zk94Il2ANZ

`Marshal()`JSONetcd。 `uuid` `json.Marshal()`。 `json.MarshalJSON()`

```
type MyStruct struct {
    uuid string
    Name string
}

func (m MyStruct) MarshalJSON() ([]byte, error {
    j, err := json.Marshal(struct {
        Uuid string
        Name string
    }{
        Uuid: m.uuid,
        Name: m.Name,
    })
```

```
    if err != nil {
           return nil, err
    }
    return j, nil
}
```

https //play.golang.org/p/Bv2k9GgbzE

Go/

structCity

```
type City struct {
    Name string
    Temperature int
}
```

encoding/jsonCity/。

GostructJSON。

```
type City struct {
    Name string `json:"name"`
    Temperature int `json:"temp"`
    // IMPORTANT: only exported fields will be encoded/decoded
    // Any field starting with a lower letter will be ignored
}
```

。 json: metadata 。

l。 t emperature json。

Cityjson.Marshal

```
// data to encode
city := City{Name: "Rome", Temperature: 30}

// encode the data
bytes, err := json.Marshal(city)
if err != nil {
    panic(err)
}

fmt.Println(string(bytes))
// {"name":"Rome","temp":30}
```

Cityjson.Unmarshal

```
// data to decode
bytes := []byte(`{"name":"Rome","temp":30}`)

// initialize the container for the decoded data
var city City
```

```
// decode the data
// notice the use of &city to pass the pointer to city
if err := json.Unmarshal(bytes, &city); err != nil {
    panic(err)
}

fmt.Println(city)
// {Rome 30}
```

JSON https://riptutorial.com/zh-CN/go/topic/994/json

13: SQL

SQLGo wiki*SQLDrivers* 。

SQL_。

Examples

`database/sql`MySql。

```
package main

import (
    "log"
    "fmt"
    "database/sql"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
    dsn := "mysql_username:CHANGEME@tcp(localhost:3306)/dbname"

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        log.Fatal(err)
    }
    defer sql.Close()

    rows, err := db.Query("select id, first_name from user limit 10")
    if err != nil {
        log.Fatal(err)
    }
    defer rows.Close()

    for rows.Next() {
        var id int
        var username string
        if err := rows.Scan(&id, &username); err != nil {
            log.Fatal(err)
        }
        fmt.Printf("%d-%s\n", id, username)
    }
}
```

MySQL

MySQL。 github.com/go-sql-driver/mysql 。

```
import (
    "database/sql"
    _ "github.com/go-sql-driver/mysql"
)
```

。

Sqlite 3

```
file := "path/to/file"
db_, err := sql.Open("sqlite3", file)
if err != nil {
    panic(err)
}
```

MySQL

```
dsn := "mysql_username:CHANGEME@tcp(localhost:3306)/dbname"
db, err := sql.Open("mysql", dsn)
if err != nil {
    panic(err)
}
```

MongoDB

```
package main

import (
    "fmt"
    "time"

    log "github.com/Sirupsen/logrus"
    mgo "gopkg.in/mgo.v2"
    "gopkg.in/mgo.v2/bson"
)

var mongoConn *mgo.Session

type MongoDB_Conn struct {
    Host string `json:"Host"`
    Port string `json:"Port"`
    User string `json:"User"`
    Pass string `json:"Pass"`
    DB   string `json:"DB"`
}

func MongoConn(mdb MongoDB_Conn) (*mgo.Session, string, error) {
    if mongoConn != nil {
        if mongoConn.Ping() == nil {
            return mongoConn, nil
        }
    }
    user := mdb.User
    pass := mdb.Pass
    host := mdb.Host
    port := mdb.Port
    db := mdb.DB
    if host == "" || port == "" || db == "" {
        log.Fatal("Host or port or db is nil")
    }
    url := fmt.Sprintf("mongodb://%s:%s@%s:%s/%s", user, pass, host, port, db)
    if user == "" {
```

```
        url = fmt.Sprintf("mongodb://%s:%s/%s", host, port, db)
    }
    mongo, err := mgo.DialWithTimeout(url, 3*time.Second)
    if err != nil {
        log.Errorf("Mongo Conn Error: [%v], Mongo ConnUrl: [%v]",
            err, url)
        errTextReturn := fmt.Sprintf("Mongo Conn Error: [%v]", err)
        return &mgo.Session{}, errors.New(errTextReturn)
    }
    mongoConn = mongo
    return mongoConn, nil
}

func MongoInsert(dbName, C string, data interface{}) error {
    mongo, err := MongoConn()
    if err != nil {
        log.Error(err)
        return err
    }
    db := mongo.DB(dbName)
    collection := db.C(C)
    err = collection.Insert(data)
    if err != nil {
        return err
    }
    return nil
}

func MongoRemove(dbName, C string, selector bson.M) error {
    mongo, err := MongoConn()
    if err != nil {
        log.Error(err)
        return err
    }
    db := mongo.DB(dbName)
    collection := db.C(C)
    err = collection.Remove(selector)
    if err != nil {
        return err
    }
    return nil
}

func MongoFind(dbName, C string, query, selector bson.M) ([]interface{}, error) {
    mongo, err := MongoConn()
    if err != nil {
        return nil, err
    }
    db := mongo.DB(dbName)
    collection := db.C(C)
    result := make([]interface{}, 0)
    err = collection.Find(query).Select(selector).All(&result)
    return result, err
}

func MongoUpdate(dbName, C string, selector bson.M, update interface{}) error {
    mongo, err := MongoConn()
    if err != nil {
        log.Error(err)
        return err
    }
```

```
    db := mongo.DB(dbName)
    collection := db.C(C)
    err = collection.Update(selector, update)
    if err != nil {
        return err
    }
    return nil
}
```

SQL https://riptutorial.com/zh-CN/go/topic/1273/sql

14: Vendoring

Go。

Go`$GOPATH/src/。 vendor 。 。`

Go 1.6。 Go 1.5`GO15VENDOREXPERIMENT=1`。

Examples

govendor

Govendorgolangvendoring。

`bosun.org/slog`

```
package main

import "bosun.org/slog"

func main() {
    slog.Infof("Hello World")
}
```

```
$GOPATH/src/
├── github.com/me/helloworld/
|   ├── hello.go
├── bosun.org/slog/
|   ├── ... (slog files)
```

`github.com/me/helloworld$GOPATH/src/bosun.org/slog/。`

Go

```
govendor add +e
```

govendor。

```
$GOPATH/src/
├── github.com/me/helloworld/
|   ├── vendor/
|   |   ├── bosun.org/slog/
|   |   |   ├── ... (slog files)
|   ├── hello.go
```

。

./vendor

trashvendor.conf。 trash

```
# package
github.com/rancher/trash

github.com/Sirupsen/logrus                      v0.10.0
github.com/urfave/cli                           v1.18.0
github.com/cloudfoundry-incubator/candiedyaml   99c3df8
https://github.com/imikushin/candiedyaml.git
github.com/stretchr/testify                     v1.1.3
github.com/davecgh/go-spew                      5215b55
github.com/pmezard/go-difflib                   792786c
golang.org/x/sys                                a408501
```

./vendor for。

#。

。 “root”。

URL。

./vendorvendor.conf

```
$ trash
```

~/.trash-cache ./vendor dir 。 ./vendor。

v0.2.5LinuxmacOSgitgitgo get。

golang / dep

golang / dep。 。 **Alpha** 。

```
$ go get -u github.com/golang/dep/...
```

```
$ dep init
$ dep ensure -update
```

```
$ dep ensure github.com/pkg/errors@^0.8.0
```

。 。

vendor.jsonGovendor

```
# It creates vendor folder and vendor.json inside it
govendor init

# Add dependencies in vendor.json
govendor fetch <dependency>
```

```
# Usage on new repository
# fetch depenencies in vendor.json
govendor sync
```

vendor.json

```
{

"comment": "",
"ignore": "test",
"package": [
    {
        "checksumSHA1": "kBeNcaKk56FguvPSUCEaH6AxpRc=",
        "path": "github.com/golang/protobuf/proto",
        "revision": "2bba0603135d7d7f5cb73b2125beeda19c09f4ef",
        "revisionTime": "2017-03-31T03:19:02Z"
    },
    {
        "checksumSHA1": "1DRAxdlWzS4U0xKN/yQ/fdNN7f0=",
        "path": "github.com/syndtr/goleveldb/leveldb/errors",
        "revision": "8c81ea47d4c41a385645e133e15510fc6a2a74b4",
        "revisionTime": "2017-04-09T01:48:31Z"
    }
],
"rootPath": "github.com/sample"

}
```

Vendoring https://riptutorial.com/zh-CN/go/topic/978/vendoring

15: XML

encoding/xmlGo struct。

XMLXML。

GobJSON。/XML。

Examples

/

XML/。 ,attr,chardata。

```
var doc = `
<parent>
  <child1 attr1="attribute one"/>
  <child2>and some cdata</child2>
</parent>
`

type parent struct {
    Child1 child1 `xml:"child1"`
    Child2 child2 `xml:"child2"`
}

type child1 struct {
    Attr1 string `xml:"attr1,attr"`
}

type child2 struct {
    Cdata1 string `xml:",cdata"`
}

func main() {
    var obj parent
    err := xml.Unmarshal([]byte(doc), &obj)
    if err != nil {
        log.Fatal(err)
    }

    fmt.Println(obj.Child2.Cdata1)

}
```

Playground

XML https://riptutorial.com/zh-CN/go/topic/1846/xml

16: YAML

Examples

YAML

```
import (
    "io/ioutil"
    "path/filepath"

    "gopkg.in/yaml.v2"
)

func main() {
    filename, _ := filepath.Abs("config/config.yml")
    yamlFile, err := ioutil.ReadFile(filename)
    var config Config
    err = yaml.Unmarshal(yamlFile, &config)
    if err != nil {
        panic(err)
    }
    //env can be accessed from config.Env
}

type Config struct {
    Env                 string `yaml:"env"`
}

//config.yml should be placed in config/config.yml for example, and needs to have the
following line for the above example:
//env: test
```

YAML https://riptutorial.com/zh-CN/go/topic/2503/yaml

17:

- CancelFunc func
- func Background
- func TODO
- func WithCancelctxCancelFunc
- func WithDeadlineContext.TimeContextCancelFunc
- func WithTimeoutContexttimeout time.DurationContextCancelFunc
- func WithValueContextkey interface {}val interface {}

`context`Go 1.7`golang.org/x/net/context`Pre 1.7API。 。

“”。 `context.Context`。

。 `context.Context` 。 。 ;。

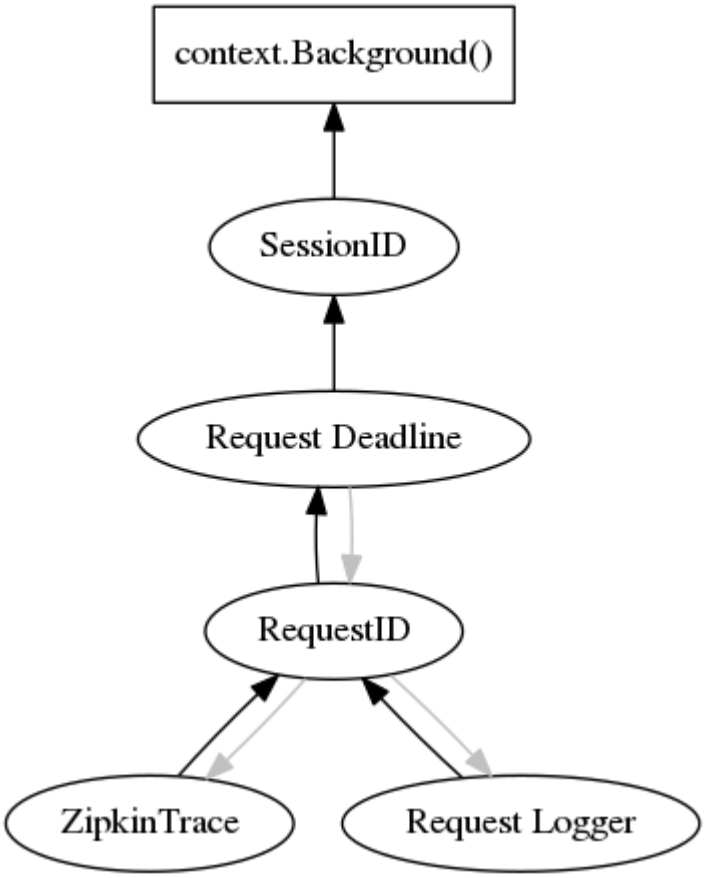
`ctx context.Context` 。 。 ContextGo。 goroutine。

- https://blog.golang.org/context

Examples

Go

```
// Pseudo-Go
ctx := context.WithValue(
    context.WithDeadline(
        context.WithValue(context.Background(), sidKey, sid),
        time.Now().Add(30 * time.Minute),
    ),
    ridKey, rid,
)
trCtx := trace.NewContext(ctx, tr)
logCtx := myRequestLogging.NewContext(ctx, myRequestLogging.NewLogger())
```



。 。 。

```
ctx, _ := context.WithTimeout(context.Background(), 200*time.Millisecond)
for {
    select {
    case <-ctx.Done():
        return ctx.Err()
    default:
        // Do an iteration of some long running work here!
    }
}
```

https://riptutorial.com/zh-CN/go/topic/2743/

18:

Go。 。 。 Ex。

http.ResponseWriter* http.Request**http.handlerFunc**。

Examples

```
func loginHandler(w http.ResponseWriter, r *http.Request) {
            // Steps to login
}


func main() {
    http.HandleFunc("/login", loginHandler)
    http.ListenAndServe(":8080", nil)
}
```

handlerFunc

```
// logger middlerware that logs time taken to process each request
func Logger(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        startTime := time.Now()
        h.ServeHttp(w,r)
        endTime := time.Since(startTime)
        log.Printf("%s %d %v", r.URL, r.Method, endTime)
    })
}

func loginHandler(w http.ResponseWriter, r *http.Request) {
            // Steps to login
}


func main() {
    http.HandleFunc("/login", Logger(loginHandler))
    http.ListenAndServe(":8080", nil)
}
```

CORS

```
func CORS(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        origin := r.Header.Get("Origin")
        w.Header().Set("Access-Control-Allow-Origin", origin)
        if r.Method == "OPTIONS" {
            w.Header().Set("Access-Control-Allow-Credentials", "true")
            w.Header().Set("Access-Control-Allow-Methods", "GET,POST")

            w.RespWriter.Header().Set("Access-Control-Allow-Headers", "Content-Type, X-CSRF-
Token, Authorization")
```

```
            return
        } else {
            h.ServeHTTP(w, r)
        }
    })
}

func main() {
    http.HandleFunc("/login", Logger(CORS(loginHandler)))
    http.ListenAndServe(":8080", nil)
}
```

Auth

```
func Authenticate(h http.Handler) http.Handler {
    return CustomHandlerFunc(func(w *http.ResponseWriter, r *http.Request) {
        // extract params from req
        // post params | headers etc
        if CheckAuth(params) {
            log.Println("Auth Pass")
            // pass control to next middleware in chain or handler func
            h.ServeHTTP(w, r)
        } else {
            log.Println("Auth Fail")
            // Responsd Auth Fail
        }
    })
}
```

```
func Recovery(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request){
        defer func() {
            if err := recover(); err != nil {
                // respondInternalServerError
            }
        }()
        h.ServeHTTP(w , r)
    })
}
```

https://riptutorial.com/zh-CN/go/topic/9343/

19:

◦ GoUTF-8◦

- variableName=“Hello World”//
- variableName=`Hello World` //
- variableName=“Hello”+“World”//
- substring=“Hello World”[04] //
- letter=“Hello World”[6] //
- fmt.Sprintf“s”“Hello World”//

Examples

string◦ ◦ ◦

```
text := "Hello World"
```

GoUTF-8◦ UTF-8UTF-8◦

"" ◦

+◦

```
text := "Hello " + "World"
```

``◦ ◦

```
text1 := "Hello\nWorld"
text2 := `Hello
World`
```

text1\ntext2◦ text1 == text2 true ◦

text2 := `Hello\nWorld`\nHello\nWorld◦ text1 := "Hello\\nWorld" ◦

fmt◦ ◦

```
%v    // the value in a default format
      // when printing structs, the plus flag (%+v) adds field names
%#v   // a Go-syntax representation of the value
%T    // a Go-syntax representation of the type of the value
%%    // a literal percent sign; consumes no value
```

```
%t    // the word true or false
```

```
%b    // base 2
```

```
%c    // the character represented by the corresponding Unicode code point
%d    // base 10
%o    // base 8
%q    // a single-quoted character literal safely escaped with Go syntax.
%x    // base 16, with lower-case letters for a-f
%X    // base 16, with upper-case letters for A-F
%U    // Unicode format: U+1234; same as "U+%04X"
```

```
%b    // decimalless scientific notation with exponent a power of two,
      // in the manner of strconv.FormatFloat with the 'b' format,
      // e.g. -123456p-78
%e    // scientific notation, e.g. -1.234456e+78
%E    // scientific notation, e.g. -1.234456E+78
%f    // decimal point but no exponent, e.g. 123.456
%F    // synonym for %f
%g    // %e for large exponents, %f otherwise
%G    // %E for large exponents, %F otherwise
```

```
%s    // the uninterpreted bytes of the string or slice
%q    // a double-quoted string safely escaped with Go syntax
%x    // base 16, lower-case, two characters per byte
%X    // base 16, upper-case, two characters per byte
```

```
%p    // base 16 notation, with leading 0x
```

```
text1 := fmt.Sprintf("Hello %s", "World")
text2 := fmt.Sprintf("%d + %d = %d", 2, 3, 5)
text3 := fmt.Sprintf("%s, %s (Age: %d)", "Obama", "Barack", 55)
```

Sprintf◦ Sprintf Printf◦

- strings.Contains

  ```
  fmt.Println(strings.Contains("foobar", "foo")) // true
  fmt.Println(strings.Contains("foobar", "baz")) // false
  ```

- strings.HasPrefix

  ```
  fmt.Println(strings.HasPrefix("foobar", "foo")) // true
  fmt.Println(strings.HasPrefix("foobar", "baz")) // false
  ```

- strings.HasSuffix

  ```
  fmt.Println(strings.HasSuffix("foobar", "bar")) // true
  fmt.Println(strings.HasSuffix("foobar", "baz")) // false
  ```

- strings.Join

  ```
  ss := []string{"foo", "bar", "bar"}
  fmt.Println(strings.Join(ss, ", ")) // foo, bar, baz
  ```

- strings.Replace

```
fmt.Println(strings.Replace("foobar", "bar", "baz", 1)) // foobaz
```

- strings.Split

```
s := "foo, bar, bar"
fmt.Println(strings.Split(s, ", ")) // [foo bar baz]
```

- strings.ToLower

```
fmt.Println(strings.ToLower("FOOBAR")) // foobar
```

- strings.ToUpper

```
fmt.Println(strings.ToUpper("foobar")) // FOOBAR
```

- strings.TrimSpace

```
fmt.Println(strings.TrimSpace("  foobar  ")) // foobar
```

https //golang.org/pkg/strings/。

https://riptutorial.com/zh-CN/go/topic/9666/

20:

Examples

Gogoroutine

```
import "sync"

func mutexTest() {
    lock := sync.Mutex{}
    go func(m *sync.Mutex) {
        m.Lock()
        defer m.Unlock()    // Automatically unlock when this function returns
        // Do some things
    }(&lock)

    lock.Lock()
    // Do some other things
    lock.Unlock()
}
```

Mutex。 Mutex.Unlock()。 Mutex.Lock();。

Mutex。 。 Go<1.7

https://riptutorial.com/zh-CN/go/topic/2607/

21:

Go。。

- GOOS = linux GOARCH = amd64 go build

$ GOOS	$ GOARCH
	386
	AMD64
	arm64
	AMD64
FreeBSD	386
FreeBSD	AMD64
FreeBSD	
Linux	386
Linux	AMD64
Linux	
Linux	arm64
Linux	PPC64
Linux	ppc64le
Linux	MIPS64
Linux	mips64le
NetBSD	386
NetBSD	AMD64
NetBSD	
OpenBSD	386
OpenBSD	AMD64

$ GOOS	$ GOARCH
OpenBSD	
Plan9	386
Plan9	AMD64
solaris	AMD64
	386
	AMD64

Examples

Makefile

MakefileWindowsMacLinuxARMx86。

```
# Replace demo with your desired executable name
appname := demo

sources := $(wildcard *.go)

build = GOOS=$(1) GOARCH=$(2) go build -o build/$(appname)$(3)
tar = cd build && tar -cvzf $(1)_$(2).tar.gz $(appname)$(3) && rm $(appname)$(3)
zip = cd build && zip $(1)_$(2).zip $(appname)$(3) && rm $(appname)$(3)

.PHONY: all windows darwin linux clean

all: windows darwin linux

clean:
    rm -rf build/

##### LINUX BUILDS #####
linux: build/linux_arm.tar.gz build/linux_arm64.tar.gz build/linux_386.tar.gz
build/linux_amd64.tar.gz

build/linux_386.tar.gz: $(sources)
    $(call build,linux,386,)
    $(call tar,linux,386)

build/linux_amd64.tar.gz: $(sources)
    $(call build,linux,amd64,)
    $(call tar,linux,amd64)

build/linux_arm.tar.gz: $(sources)
    $(call build,linux,arm,)
    $(call tar,linux,arm)

build/linux_arm64.tar.gz: $(sources)
    $(call build,linux,arm64,)
    $(call tar,linux,arm64)
```

```
##### DARWIN (MAC) BUILDS #####
darwin: build/darwin_amd64.tar.gz

build/darwin_amd64.tar.gz: $(sources)
    $(call build,darwin,amd64,)
    $(call tar,darwin,amd64)

##### WINDOWS BUILDS #####
windows: build/windows_386.zip build/windows_amd64.zip

build/windows_386.zip: $(sources)
    $(call build,windows,386,.exe)
    $(call zip,windows,386,.exe)

build/windows_amd64.zip: $(sources)
    $(call build,windows,amd64,.exe)
    $(call zip,windows,amd64,.exe)
```

Makefile

go build

`go buildGOOSGOARCH`

Mac64

```
GOOS=darwin GOARCH=amd64 go build
```

Windows x86

```
GOOS=windows GOARCH=386 go build
```

```
GOOS=windows GOARCH=386 go build -o appname_win_x86.exe
```

1.7GOOSGOARCH

```
go tool dist list
```

`go tool dist list -json`

gox

`gox` https //github.com/mitchellh/gox

`go get github.com/mitchellh/gox`。 Go`/golang/bin~/golang/bin` 。 `gox`。

Go`go build gox`x86ARMLinuxmacOSWindows。

gox -os="linux" ◦ gox -osarch="linux/amd64" ◦

Linuxarmhelloworld.go

helloworld.go

```
package main

import "fmt"

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

```
GOOS=linux GOARCH=arm go build helloworld.go
```

helloworld arm◦

https://riptutorial.com/zh-CN/go/topic/1020/

22: AtomGo

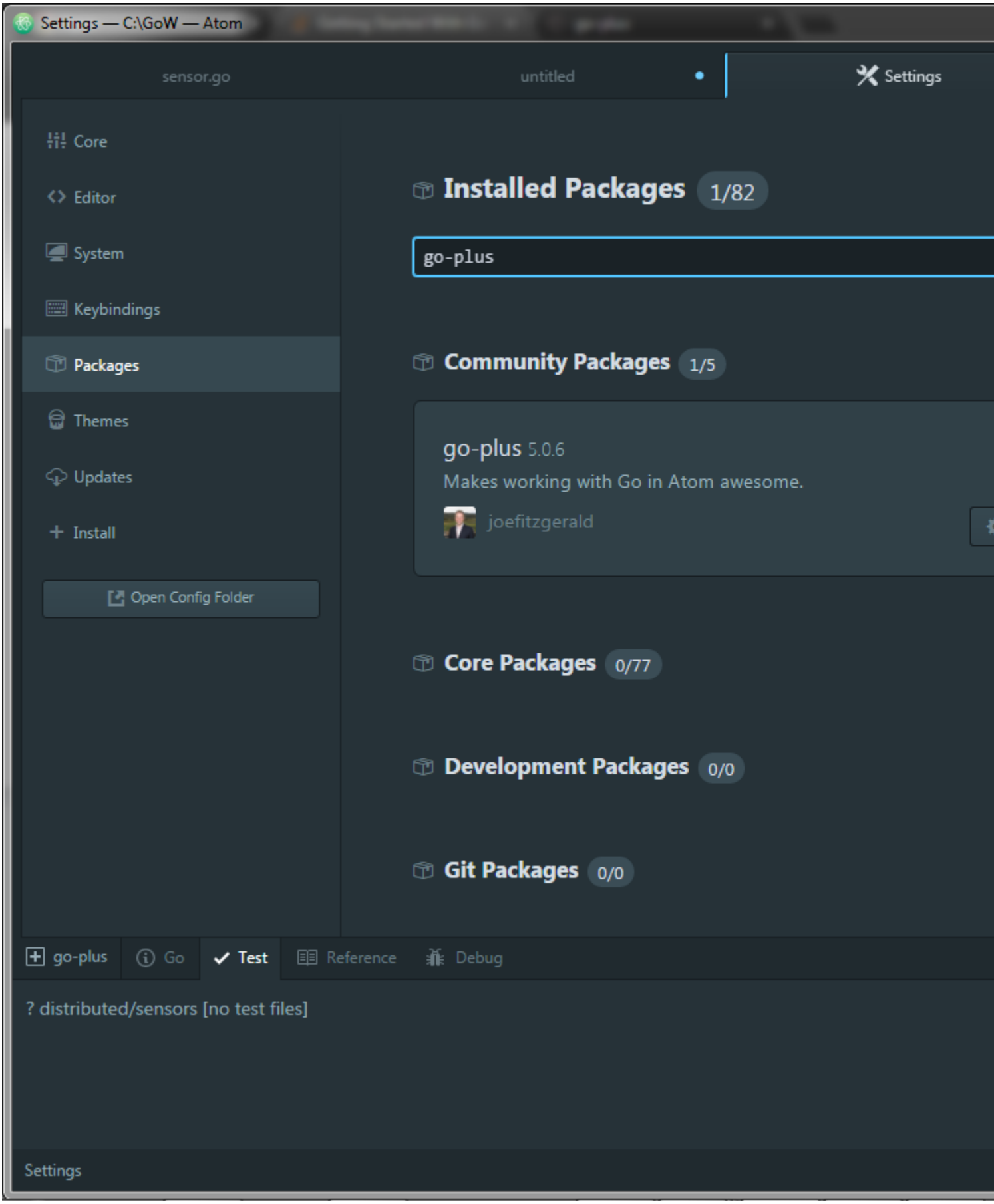
http://www.riptutorial.com/go/topic/198/getting-started-with-go 。 Atom https://atom.io gulp。 *gulp*。

Examples

AtomGulp

1. Atom。
2. Atomctrl +。 - >go-plus go-plus

Atomgo-plus



3. go get:(

   go get -u golang.org/x/tools/cmd/goimports

go get -u golang.org/x/tools/cmd/gorename

-u github.com/sqs/goreturns

-u github.com/nsf/gocode

-u github.com/alecthomas/gometalinter

-u github.com/zmb3/gogetdoc

-u github.com/rogpeppe/godef

go get -u golang.org/x/tools/cmd/guru

4. npm gulp-getting-started-doc Gulp Gulpjs

   $ npm install --global gulp

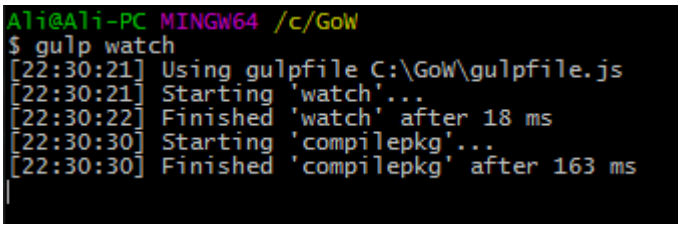
$ GO_PATH / gulpfile.js

```
var gulp = require('gulp');
var path = require('path');
var shell = require('gulp-shell');

var goPath = 'src/mypackage/**/*.go';


gulp.task('compilepkg', function() {
  return gulp.src(goPath, {read: false})
    .pipe(shell(['go install <%= stripPath(file.path) %>'],
      {
          templateData: {
            stripPath: function(filePath) {
              var subPath = filePath.substring(process.cwd().length + 5);
              var pkg = subPath.substring(0, subPath.lastIndexOf(path.sep));
              return pkg;
            }
          }
      })
    );
});

gulp.task('watch', function() {
  gulp.watch(goPath, ['compilepkg']);
});
```

*compliepkg*goPathsrc / mypackage /go。shellgo install changed_file.go

gogulp

gulp

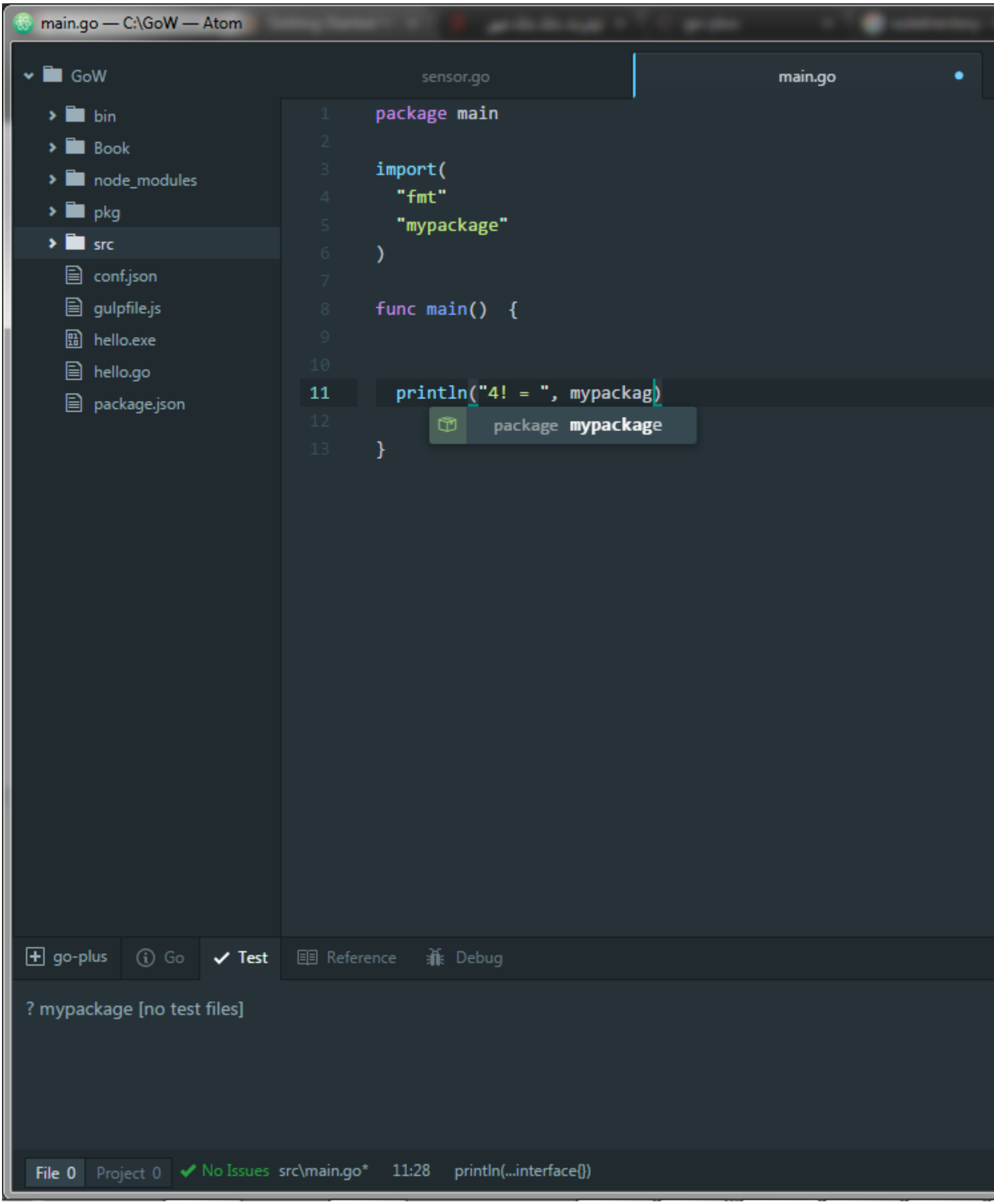


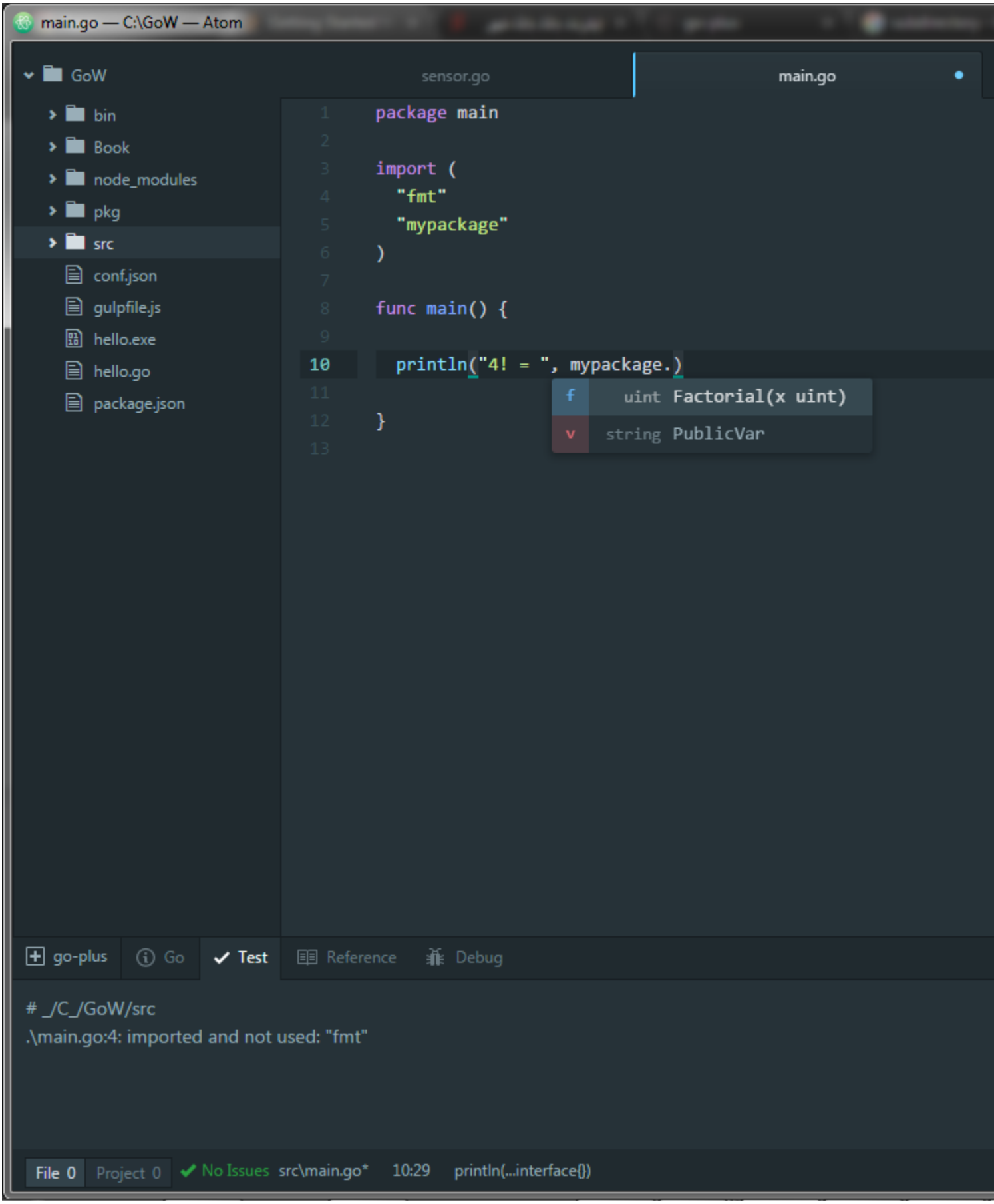
$ GO_PATH / mypackage / source.go

```
package mypackage

var PublicVar string = "Hello, dear reader!"

//Calculates the factorial of given number recursively!
func Factorial(x uint) uint {
    if x == 0 {
        return 1
    }
    return x * Factorial(x-1)
}
```

$ GO_PATH / main.go

AtomGulpgo


main.go — C:\GoW — Atom
GoW
bin
Book
node_modules
pkg
src
conf.json
gulpfile.js
hello.exe
hello.go
package.json
sensor.go
main.go
1 package main
2
3 import(
4 "fmt"
5 "mypackage"
6)
7
8 func main() {
9
10
11 println("4! = ", mypackag)
12
13 }
package mypackage
go-plus
Go
Test
Reference
Debug
? mypackage [no test files]
File 0
Project 0
No Issues
src\main.go*
11:28
println(...interface{})




```
package main

import (
    "fmt"
```

```
    "mypackage"
)

func main() {

    println("4! = ", mypackage.Factorial(4))

}
```

```
Ali@Ali-PC MINGW64 /c/GoW
$ go run src/main.go
4! =  24
```

AtomGo https://riptutorial.com/zh-CN/go/topic/8592/atomgo

23: gopprof

go ◦

Examples

cpu

◦

```
var cpuprofile = flag.String("cpuprofile", "", "write cpu profile `file`")
var memprofile = flag.String("memprofile", "", "write memory profile to `file`")

func main() {
    flag.Parse()
    if *cpuprofile != "" {
        f, err := os.Create(*cpuprofile)
        if err != nil {
            log.Fatal("could not create CPU profile: ", err)
        }
        if err := pprof.StartCPUProfile(f); err != nil {
            log.Fatal("could not start CPU profile: ", err)
        }
        defer pprof.StopCPUProfile()
    }
    ...
    if *memprofile != "" {
        f, err := os.Create(*memprofile)
        if err != nil {
            log.Fatal("could not create memory profile: ", err)
        }
        runtime.GC() // get up-to-date statistics
        if err := pprof.WriteHeapProfile(f); err != nil {
            log.Fatal("could not write memory profile: ", err)
        }
        f.Close()
    }
}
```

gomain `go build main.go◦ main.exe -cpuprofile cpu.prof -memprof mem.prof◦ go test -cpuprofile cpu.prof -memprofile mem.prof`

```
var memprofile = flag.String("memprofile", "", "write memory profile to `file`")

func main() {
    flag.Parse()
    if *memprofile != "" {
        f, err := os.Create(*memprofile)
        if err != nil {
            log.Fatal("could not create memory profile: ", err)
        }
        runtime.GC() // get up-to-date statistics
        if err := pprof.WriteHeapProfile(f); err != nil {
            log.Fatal("could not write memory profile: ", err)
```

```
        }
        f.Close()
    }
}
```

```
go build main.go
main.exe -memprofile mem.prof
go tool pprof main.exe mem.prof
```

CPU /

```
// Sets the CPU profiling rate to hz samples per second
// If hz <= 0, SetCPUProfileRate turns off profiling
runtime.SetCPUProfileRate(hz)

// Controls the fraction of goroutine blocking events that are reported in the blocking
profile
// Rate = 1 includes every blocking event in the profile
// Rate <= 0 turns off profiling
runtime.SetBlockProfileRate(rate)
```

main

```
func BenchmarkHello(b *testing.B) {
    for i := 0; i < b.N; i++ {
        fmt.Sprintf("hello")
    }
}
```

go test -cpuprofile cpu.prof -bench =。

cpu.profprof。

prof**go**prof

pprof cpu.prof

profile

```
(pprof) top
```

```
(pprof) peek
```

。

```
(pprof) web
```

svg。

top

```
69.29s of 100.84s total (68.71%)
Dropped 176 nodes (cum <= 0.50s)
Showing top 10 nodes out of 73 (cum >= 12.03s)
      flat  flat%   sum%        cum   cum%
    12.44s 12.34% 12.34%     27.87s 27.64%  runtime.mapaccess1
    10.94s 10.85% 23.19%     10.94s 10.85%  runtime.duffcopy
     9.45s  9.37% 32.56%     54.61s 54.16%  github.com/tester/test.(*Circle).Draw
     8.88s  8.81% 41.36%      8.88s  8.81%  runtime.aeshashbody
     7.90s  7.83% 49.20%     11.04s 10.95%  runtime.mapaccess1_fast64
     5.86s  5.81% 55.01%      9.59s  9.51%  github.com/tester/test.(*Circle).isCircle
     5.03s  4.99% 60.00%      8.89s  8.82%  github.com/tester/test.(*Circle).openCircle
     3.14s  3.11% 63.11%      3.14s  3.11%  runtime.aeshash64
     3.08s  3.05% 66.16%      7.85s  7.78%  runtime.mallocgc
     2.57s  2.55% 68.71%     12.03s 11.93%  runtime.memhash
```

gopprof https://riptutorial.com/zh-CN/go/topic/7748/gopprof

24:

Examples

◦ ◦ ◦

```
func g() {
    i := 0
    f := func() { // anonymous function
        fmt.Println("f called")
    }
}
```

g f fg◦ gf gf ◦

```
func NaturalNumbers() func() int {
    i := 0
    f:= func() int { // f is the function part of closure
        i++
        return i
    }
    return f
}
```

NaturalNumbersNaturalNumbersf ◦ f NaturalNumbersi ◦

NaturalNumbers NaturalNumbers

```
n := NaturalNumbers()
```

n◦ f NaturalNumbers◦

n i

n

```
fmt.Println(n()) // 1
fmt.Println(n()) // 2
fmt.Println(n()) // 3
```

n i ◦ i0ni++ ◦

i◦ ◦

NaturalNumbers◦ NaturalNumbersi ◦ f i ◦

```
o := NaturalNumbers()

fmt.Println(n()) // 4
fmt.Println(o()) // 1
```

```
fmt.Println(o()) // 2
fmt.Println(n()) // 5
```

no◦ ◦

```
func multiples(i int) func() int {
    var x int = 0
    return func() int {
        x++
        // paramenter to multiples (here it is i) also forms
        // a part of the environment, and is retained
        return x * i
    }
}

two := multiples(2)
fmt.Println(two(), two(), two()) // 2 4 6

fortyTwo := multiples(42)
fmt.Println(fortyTwo(), fortyTwo(), fortyTwo()) // 42 84 126
```

https://riptutorial.com/zh-CN/go/topic/2741/

25:

sync.Pool。 。

Examples

sync.Pool

sync.Poolsync.Pool。

```
package main

import (
    "bytes"
    "fmt"
    "sync"
)

var pool = sync.Pool{
    // New creates an object when the pool has nothing available to return.
    // New must return an interface{} to make it flexible. You have to cast
    // your type after getting it.
    New: func() interface{} {
        // Pools often contain things like *bytes.Buffer, which are
        // temporary and re-usable.
        return &bytes.Buffer{}
    },
}

func main() {
    // When getting from a Pool, you need to cast
    s := pool.Get().(*bytes.Buffer)
    // We write to the object
    s.Write([]byte("dirty"))
    // Then put it back
    pool.Put(s)

    // Pools can return dirty results

    // Get 'another' buffer
    s = pool.Get().(*bytes.Buffer)
    // Write to it
    s.Write([]bytes("append"))
    // At this point, if GC ran, this buffer *might* exist already, in
    // which case it will contain the bytes of the string "dirtyappend"
    fmt.Println(s)
    // So use pools wisely, and clean up after yourself
    s.Reset()
    pool.Put(s)

    // When you clean up, your buffer should be empty
    s = pool.Get().(*bytes.Buffer)
    // Defer your Puts to make sure you don't leak!
    defer pool.Put(s)
    s.Write([]byte("reset!"))
    // This prints "reset!", and not "dirtyappendreset!"
```

```
    fmt.Println(s)
}
```

https://riptutorial.com/zh-CN/go/topic/4647/

26:

。;/。CALLPUSHGOTOx86。。

Go inlines。goroutine。

- ... args
- “”
- `panic recoverdefer`

Examples

`go:noinline` pragma。

```
package main

func printhello() {
    println("Hello")
}

func main() {
    printhello()
}
```

```
$ go version
go version go1.6.2 linux/amd64
$ go build main.go
$ ./main
Hello
$ go tool objdump main
TEXT main.main(SB) /home/sam/main.go
        main.go:7       0x401000        64488b0c25f8ffffff      FS MOVQ FS:0xfffffff8, CX
        main.go:7       0x401009        483b6110                CMPQ 0x10(CX), SP
        main.go:7       0x40100d        7631                    JBE 0x401040
        main.go:7       0x40100f        4883ec10                SUBQ $0x10, SP
        main.go:8       0x401013        e8281f0200              CALL runtime.printlock(SB)
        main.go:8       0x401018        488d1d01130700          LEAQ 0x71301(IP), BX
        main.go:8       0x40101f        48891c24                MOVQ BX, 0(SP)
        main.go:8       0x401023        48c744240805000000      MOVQ $0x5, 0x8(SP)
        main.go:8       0x40102c        e81f290200              CALL runtime.printstring(SB)
        main.go:8       0x401031        e89a210200              CALL runtime.printnl(SB)
        main.go:8       0x401036        e8851f0200              CALL runtime.printunlock(SB)
        main.go:9       0x40103b        4883c410                ADDQ $0x10, SP
        main.go:9       0x40103f        c3                      RET
        main.go:7       0x401040        e87b9f0400              CALL
runtime.morestack_noctxt(SB)
        main.go:7       0x401045        ebb9                    JMP main.main(SB)
        main.go:7       0x401047        cc                      INT $0x3
        main.go:7       0x401048        cc                      INT $0x3
        main.go:7       0x401049        cc                      INT $0x3
        main.go:7       0x40104a        cc                      INT $0x3
        main.go:7       0x40104b        cc                      INT $0x3
        main.go:7       0x40104c        cc                      INT $0x3
        main.go:7       0x40104d        cc                      INT $0x3
```

```
        main.go:7       0x40104e        cc                      INT $0x3
        main.go:7       0x40104f        cc                      INT $0x3
…
```

`printhello` `CALL` ◦ pragma

```
package main

//go:noinline
func printhello() {
    println("Hello")
}

func main() {
    printhello()
}
```

printhello`CALL main.printhello`

```
$ go version
go version go1.6.2 linux/amd64
$ go build main.go
$ ./main
Hello
$ go tool objdump main
TEXT main.printhello(SB) /home/sam/main.go
        main.go:4       0x401000        64488b0c25f8ffffff      FS MOVQ FS:0xfffffff8, CX
        main.go:4       0x401009        483b6110                CMPQ 0x10(CX), SP
        main.go:4       0x40100d        7631                    JBE 0x401040
        main.go:4       0x40100f        4883ec10                SUBQ $0x10, SP
        main.go:5       0x401013        e8481f0200              CALL runtime.printlock(SB)
        main.go:5       0x401018        488d1d01130700          LEAQ 0x71301(IP), BX
        main.go:5       0x40101f        48891c24                MOVQ BX, 0(SP)
        main.go:5       0x401023        48c744240805000000      MOVQ $0x5, 0x8(SP)
        main.go:5       0x40102c        e83f290200              CALL runtime.printstring(SB)
        main.go:5       0x401031        e8ba210200              CALL runtime.printnl(SB)
        main.go:5       0x401036        e8a51f0200              CALL runtime.printunlock(SB)
        main.go:6       0x40103b        4883c410                ADDQ $0x10, SP
        main.go:6       0x40103f        c3                      RET
        main.go:4       0x401040        e89b9f0400              CALL
runtime.morestack_noctxt(SB)
        main.go:4       0x401045        ebb9                    JMP main.printhello(SB)
        main.go:4       0x401047        cc                      INT $0x3
        main.go:4       0x401048        cc                      INT $0x3
        main.go:4       0x401049        cc                      INT $0x3
        main.go:4       0x40104a        cc                      INT $0x3
        main.go:4       0x40104b        cc                      INT $0x3
        main.go:4       0x40104c        cc                      INT $0x3
        main.go:4       0x40104d        cc                      INT $0x3
        main.go:4       0x40104e        cc                      INT $0x3
        main.go:4       0x40104f        cc                      INT $0x3

TEXT main.main(SB) /home/sam/main.go
        main.go:8       0x401050        64488b0c25f8ffffff      FS MOVQ FS:0xfffffff8, CX
        main.go:8       0x401059        483b6110                CMPQ 0x10(CX), SP
        main.go:8       0x40105d        7606                    JBE 0x401065
        main.go:9       0x40105f        e89cffffff              CALL main.printhello(SB)
        main.go:10      0x401064        c3                      RET
        main.go:8       0x401065        e8769f0400              CALL
```

```
runtime.morestack_noctxt(SB)
        main.go:8       0x40106a        ebe4                    JMP main.main(SB)
        main.go:8       0x40106c        cc                      INT $0x3
        main.go:8       0x40106d        cc                      INT $0x3
        main.go:8       0x40106e        cc                      INT $0x3
        main.go:8       0x40106f        cc                      INT $0x3
…
```

https://riptutorial.com/zh-CN/go/topic/2718/

27:

GobGo。Go。 GobXML。

。

Examples

gob

```
package main

import (
    "encoding/gob"
    "os"
)

type User struct {
    Username string
    Password string
}

func main() {

    user := User{
        "zola",
        "supersecretpassword",
    }

    file, _ := os.Create("user.gob")

    defer file.Close()

    encoder := gob.NewEncoder(file)

    encoder.Encode(user)

}
```

go

```
package main

import (
    "encoding/gob"
    "fmt"
    "os"
)

type User struct {
    Username string
    Password string
}
```

```
func main() {

    user := User{}

    file, _ := os.Open("user.gob")

    defer file.Close()

    decoder := gob.NewDecoder(file)

    decoder.Decode(&user)

    fmt.Println(user)

}
```

gob

```
package main

import (
    "encoding/gob"
    "fmt"
    "os"
)

type User struct {
    Username string
    Password string
}

type Admin struct {
    Username string
    Password string
    IsAdmin  bool
}

type Deleter interface {
    Delete()
}

func (u User) Delete() {
    fmt.Println("User ==> Delete()")
}

func (a Admin) Delete() {
    fmt.Println("Admin ==> Delete()")
}

func main() {

    user := User{
        "zola",
        "supersecretpassword",
    }

    admin := Admin{
        "john",
        "supersecretpassword",
```

```
        true,
    }

    file, _ := os.Create("user.gob")

    adminFile, _ := os.Create("admin.gob")

    defer file.Close()

    defer adminFile.Close()

    gob.Register(User{}) // registering the type allows us to encode it

    gob.Register(Admin{}) // registering the type allows us to encode it

    encoder := gob.NewEncoder(file)

    adminEncoder := gob.NewEncoder(adminFile)

    InterfaceEncode(encoder, user)

    InterfaceEncode(adminEncoder, admin)

}

func InterfaceEncode(encoder *gob.Encoder, d Deleter) {

    if err := encoder.Encode(&d); err != nil {
        fmt.Println(err)
    }

}
```

gob

```
package main

import (
    "encoding/gob"
    "fmt"
    "log"
    "os"
)

type User struct {
    Username string
    Password string
}

type Admin struct {
    Username string
    Password string
    IsAdmin  bool
}

type Deleter interface {
    Delete()
}
```

```
func (u User) Delete() {
    fmt.Println("User ==> Delete()")
}

func (a Admin) Delete() {
    fmt.Println("Admin ==> Delete()")
}

func main() {

    file, _ := os.Open("user.gob")

    adminFile, _ := os.Open("admin.gob")

    defer file.Close()

    defer adminFile.Close()

    gob.Register(User{}) // registering the type allows us to encode it

    gob.Register(Admin{}) // registering the type allows us to encode it

    var admin Deleter

    var user Deleter

    userDecoder := gob.NewDecoder(file)

    adminDecoder := gob.NewDecoder(adminFile)

    user = InterfaceDecode(userDecoder)

    admin = InterfaceDecode(adminDecoder)

    fmt.Println(user)

    fmt.Println(admin)

}

func InterfaceDecode(decoder *gob.Decoder) Deleter {

    var d Deleter

    if err := decoder.Decode(&d); err != nil {
        log.Fatal(err)
    }

    return d

}
```

https://riptutorial.com/zh-CN/go/topic/8820/

28:

Examples

switch

```
switch a + b {
case c:
    // do something
case d:
    // do something else
default:
    // do something entirely different
}
```

```
if a + b == c {
    // do something
} else if a + b == d {
    // do something else
} else {
    // do something entirely different
}
```

---

default◦

```
switch a + b {
default:
    // do something entirely different
case c:
    // do something
case d:
    // do something else
}
```

defaultfallthrough

```
switch a + b {
default:
    // do something entirely different, but then also do something
    fallthrough
case c:
    // do something
case d:
    // do something else
}
```

---

true

```
switch {
case a + b == c:
    // do something
```

```
case a + b == d:
    // do something else
}
```

Switch`if`

```
switch n := getNumber(); n {
case 1:
    // do something
case 2:
    // do something else
}
```

```
switch a + b {
case c, d:
    // do something
default:
    // do something entirely different
}
```

`if`

```
if a == b {
    // do something
}
```

`{`

```
if a == b
{
    // do something
}
```

`elseif`

```
if a == b {
    // do something
} else if a == c {
    // do something else
} else {
    // do something entirely different
}
```

---

golang.org “ ”`if`

```
if err := attemptSomething(); err != nil {
    // attemptSomething() was successful!
} else {
    // attemptSomething() returned an error; handle it
}
fmt.Println(err) // compiler error, 'undefined: err'
```

```
// assuming x is an expression of type interface{}
switch t := x.(type) {
case nil:
    // x is nil
    // t will be type interface{}
case int:
    // underlying type of x is int
    // t will be int in this case as well
case string:
    // underlying type of x is string
    // t will be string in this case as well
case float, bool:
    // underlying type of x is either float or bool
    // since we don't know which, t is of type interface{} in this case
default:
    // underlying type of x was not any of the types tested for
    // t is interface{} in this type
}
```

---

error

```
switch t := x.(type) {
case error:
    log.Fatal(t)
case myType:
    fmt.Println(myType.message)
case myInterface:
    t.MyInterfaceMethod()
case func(string) bool:
    if t("Hello world?") {
        fmt.Println("Hello world!")
    }
}
```

goto◦ gotogoto◦

https //golang.org/src/math/gamma.go

```
    for x < 0 {
        if x > -1e-09 {
            goto small
        }
        z = z / x
        x = x + 1
    }
    for x < 2 {
        if x < 1e-09 {
            goto small
        }
        z = z / x
        x = x + 1
    }

    if x == 2 {
        return z
    }
```

```
    x = x - 2
    p = (((((x*_gamP[0]+_gamP[1])*x+_gamP[2])*x+_gamP[3])*x+_gamP[4])*x+_gamP[5])*x + _gamP[6]
    q =
((((((x*_gamQ[0]+_gamQ[1])*x+_gamQ[2])*x+_gamQ[3])*x+_gamQ[4])*x+_gamQ[5])*x+_gamQ[6])*x +
_gamQ[7]
    return z * p / q

small:
    if x == 0 {
        return Inf(1)
    }
    return z / ((1 + Euler*x) * x)
```

-

break

```
import "fmt"

func main() {
    i:=0
    for true {
      if i>2 {
        break
        }
    fmt.Println("Iteration : ",i)
    i++
    }
}
```

continue

```
import "fmt"

func main() {
    j:=100
    for j<110 {
     j++
     if j%2==0 {
        continue
        }
    fmt.Println("Var : ",j)
    }
}
```

/

```
import "fmt"

func main() {
    j := 100

loop:
    for j < 110 {
        j++
```

```
        switch j % 3 {
        case 0:
            continue loop
        case 1:
            break loop
        }

        fmt.Println("Var : ", j)
    }
}
```

https://riptutorial.com/zh-CN/go/topic/1342/

29:

Go。

- “ `// +build` ”
- `!`
- OR

- `// +build`package。
- 。

FreeBSD
Linux
NetBSD
OpenBSD
Plan9
solaris

https://golang.org/doc/install/source#environment`$GOOS`。

Examples

```
// +build linux

package lib

var OnlyAccessibleInLinux int // Will only be compiled in Linux
```

`!`

```
// +build !windows

package lib

var NotWindows int // Will be compiled in all platforms but not Windows
```

```
// +build linux darwin plan9

package lib
```

```
var SomeUnix int // Will be compiled in linux, darwin and plan9 but not on others
```

`lib_linux.go` linux

```
package lib

var OnlyCompiledInLinux string
```

。 。

`main.go`

```
package main

import "fmt"

func main() {
    fmt.Println("Hello World from Conditional Compilation Doc!")
    printDetails()
}
```

`details.go`

```
// +build !windows

package main

import "fmt"

func printDetails() {
    fmt.Println("Some specific details that cannot be found on Windows")
}
```

`details_windows.go`

```
package main

import "fmt"

func printDetails() {
    fmt.Println("Windows specific details")
}
```

https://riptutorial.com/zh-CN/go/topic/8599/

30:

Go。 。 Go。

- func//
- funcx intint //
- funcab intz float32bool //2
- func... int//“variadic”
- funcintbool//
- funcab intz float64opt ... interface {}success bool//2floatboolean value

Examples

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

```
func SayHelloToMe(firstName, lastName string, age int) {
    fmt.Printf("Hello, %s %s!\n", firstName, lastName)
    fmt.Printf("You are %d", age)
}
```

firstNamelastName。

```
func AddAndMultiply(a, b int) (int, int) {
    return a+b, a*b
}
```

var

```
import errors

func Divide(dividend, divisor int) (int, error) {
    if divisor == 0 {
        return 0, errors.New("Division by zero forbidden")
    }
    return dividend / divisor, nil
}
```

- 。
- return。

。 return。 “”。

```
func Inverse(v float32) (reciprocal float32) {
    if v == 0 {
        return
    }
```

```
    reciprocal = 1 / v
    return
}
```

```
//A function can also return multiple values
func split(sum int) (x, y int) {
    x = sum * 4 / 9
    y = sum - x
    return
}
```

- 。
- return。

Hello!stdout

```
package main

import "fmt"

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

---

strstdout

```
package main

import "fmt"

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

---

str

```
package main

import "fmt"

func main() {
    str := "Hello!"
    func() {
        fmt.Println(str)
    }()
}
```

```
package main

import (
    "fmt"
)

func main() {
    str := "Hello!"
    anon := func() {
        fmt.Println(str)
    }
    anon()
}
```

。。

```
package main

import "fmt"

func variadic(strs ...string) {
     // strs is a slice of string
     for i, str := range strs {
         fmt.Printf("%d: %s\n", i, str)
     }
}

func main() {
     variadic("Hello", "Goodbye")
     variadic("Str1", "Str2", "Str3")
}
```

...

```
func main() {
     strs := []string {"Str1", "Str2", "Str3"}

     variadic(strs...)
}
```

https://riptutorial.com/zh-CN/go/topic/373/

31:

Go。Go。

Examples

Go。。Github。

Go。。secret

```
type secret struct {
    DisplayName       string
    Notes             string
    Username          string
    EMail             string
    CopyMethod        string
    Password          string
    CustomField01Name string
    CustomField01Data string
    CustomField02Name string
    CustomField02Data string
    CustomField03Name string
    CustomField03Data string
    CustomField04Name string
    CustomField04Data string
    CustomField05Name string
    CustomField05Data string
    CustomField06Name string
    CustomField06Data string
}
```

secret。Github。

1

```
masterPassword := "PASS"
```

2

。。

```
secretBytesDecrypted :=
[]byte(fmt.Sprintf("%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n",
    artifact.DisplayName,
    strings.Replace(artifact.Notes, "\n", string(65000), -1),
    artifact.Username,
    artifact.EMail,
```

```
    artifact.CopyMethod,
    artifact.Password,
    artifact.CustomField01Name,
    artifact.CustomField01Data,
    artifact.CustomField02Name,
    artifact.CustomField02Data,
    artifact.CustomField03Name,
    artifact.CustomField03Data,
    artifact.CustomField04Name,
    artifact.CustomField04Data,
    artifact.CustomField05Name,
    artifact.CustomField05Data,
    artifact.CustomField06Name,
    artifact.CustomField06Data,
))
```

3

`saltBytes := uuid.NewV4().Bytes()` 。 UUID v4。

4

RFC 2898

```
keyLength := 256
rfc2898Iterations := 6

keyVectorData := pbkdf2.Key(masterPassword, saltBytes, rfc2898Iterations,
(keyLength/8)+aes.BlockSize, sha1.New)
keyBytes := keyVectorData[:keyLength/8]
vectorBytes := keyVectorData[keyLength/8:]
```

5

CBC。 。

```
if len(secretBytesDecrypted)%aes.BlockSize != 0 {
    numberNecessaryBlocks := int(math.Ceil(float64(len(secretBytesDecrypted)) /
float64(aes.BlockSize)))
    enhanced := make([]byte, numberNecessaryBlocks*aes.BlockSize)
    copy(enhanced, secretBytesDecrypted)
    secretBytesDecrypted = enhanced
}
```

6

AES `aesBlockEncrypter, aesErr := aes.NewCipher(keyBytes)`

7

encryptedData := make([]byte, len(secretBytesDecrypted)) ◦ AES-CBC◦

8

```
aesEncrypter := cipher.NewCBCEncrypter(aesBlockEncrypter, vectorBytes)
aesEncrypter.CryptBlocks(encryptedData, secretBytesDecrypted)
```

encryptedData◦

9

◦ ◦ ◦ base64

```
encodedBytes := make([]byte, base64.StdEncoding.EncodedLen(len(encryptedData)))
base64.StdEncoding.Encode(encodedBytes, encryptedData)
```

◦ salt[0x10]base64 content ◦ ◦ base64₁₀ ◦ base64◦ 10base64◦

```
fileContent := make([]byte, len(saltBytes))
copy(fileContent, saltBytes)
fileContent = append(fileContent, 10)
fileContent = append(fileContent, encodedBytes...)
```

10

writeErr := ioutil.WriteFile("my secret.data", fileContent, 0644) ◦

◦ secret ◦ ◦

1

masterPassword := "PASS" ◦

2

encryptedFileData, bytesErr := ioutil.ReadFile(filename) ◦

3

10

```
for n := len(encryptedFileData) - 1; n > 0; n-- {
    if encryptedFileData[n] == 10 {
```

```
        saltBytes = encryptedFileData[:n]
        encryptedBytesBase64 = encryptedFileData[n+1:]
        break
    }
}
```

4

base64

```
decodedBytes := make([]byte, len(encryptedBytesBase64))
countDecoded, decodedErr := base64.StdEncoding.Decode(decodedBytes, encryptedBytesBase64)
encryptedBytes = decodedBytes[:countDecoded]
```

5

RFC 2898

```
keyLength := 256
rfc2898Iterations := 6

keyVectorData := pbkdf2.Key(masterPassword, saltBytes, rfc2898Iterations,
(keyLength/8)+aes.BlockSize, sha1.New)
keyBytes := keyVectorData[:keyLength/8]
vectorBytes := keyVectorData[keyLength/8:]
```

6

AES `aesBlockDecrypter, aesErr := aes.NewCipher(keyBytes)` ◦

7

`decryptedData := make([]byte, len(encryptedBytes))` ◦ ◦

8

```
aesDecrypter := cipher.NewCBCDecrypter(aesBlockDecrypter, vectorBytes)
aesDecrypter.CryptBlocks(decryptedData, encryptedBytes)
```

9

`decryptedString := string(decryptedData)` ◦ `lines := strings.Split(decryptedString, "\n")` ◦

10

`secret`

```
artifact := secret{}
artifact.DisplayName = lines[0]
artifact.Notes = lines[1]
artifact.Username = lines[2]
artifact.EMail = lines[3]
artifact.CopyMethod = lines[4]
artifact.Password = lines[5]
artifact.CustomField01Name = lines[6]
artifact.CustomField01Data = lines[7]
artifact.CustomField02Name = lines[8]
artifact.CustomField02Data = lines[9]
artifact.CustomField03Name = lines[10]
artifact.CustomField03Data = lines[11]
artifact.CustomField04Name = lines[12]
artifact.CustomField04Data = lines[13]
artifact.CustomField05Name = lines[14]
artifact.CustomField05Data = lines[15]
artifact.CustomField06Name = lines[16]
artifact.CustomField06Data = lines[17]
```

notes `artifact.Notes = strings.Replace(artifact.Notes, string(65000), "\n", -1)` ◦

https://riptutorial.com/zh-CN/go/topic/10065/

32:

Examples

`init` main ◦

```
package usefull

func init() {
    // init code
}
```

◦

```
import _ "usefull"
```

Go`go get <package>`global / shared `$GOPATH/src`◦ ◦ `go get`◦

`vendor/`◦ ◦

```
$GOPATH/src/
├── github.com/username/project/
|   ├── main.go
|   ├── vendor/
|   |   ├── github.com/pkg/errors
|   |   ├── github.com/gorilla/mux
```

◦ package◦

`$GOPATH/src/mypck a.go`

```
package apple

const Pi = 3.14
```

```
package main

import (
    "mypck"
    "fmt"
)

func main() {
    fmt.Println(apple.Pi)
}
```

◦

◦ Go idetifier

```
identifier = letter { letter | unicode_digit } .
```

unicodeαβGo。 Go。 Go。

。

```
import "path/to/package"
```

```
import (
    "path/to/package1"
    "path/to/package2"
)
```

import$GOPATH.gopackagename.AnyExportedName。

./。

```
project
├── src
│   ├── package1
│   │   └── file1.go
│   └── package2
│       └── file2.go
└── main.go
```

main.gofile1.gofile2.go

```
import (
    "./src/package1"
    "./src/package2"
)
```

。 import-statement。

```
import (
    "fmt" //fmt from the standardlibrary
    tfmt "some/thirdparty/fmt" //fmt from some other library
)
```

fmtfmt.* fmttfmt.* 。

package.package.

```
import (
    . "fmt"
)
```

fmtPrintf Playground

Go。

```
import (
    _ "fmt"
)
```

init。

LinuxOS-X“” - 。

“path / to / Package1”vs“path / to / package1”

https //github.com/akamai-open/AkamaiOPEN-edgegrid-golang/issues/2

https://riptutorial.com/zh-CN/go/topic/401/

33:

reflect◦ **runtime** ◦

static typeGo lang

Examples

◦

```
import "reflect"

value := reflect.ValueOf(4)

// Interface returns an interface{}-typed value, which can be type-asserted
value.Interface().(int) // 4

// Type gets the reflect.Type, which contains runtime type information about
// this value
value.Type().Name() // int

value.SetInt(5) // panics -- non-pointer/slice/array types are not addressable

x := 4
reflect.ValueOf(&x).Elem().SetInt(5) // works
```

```
import "reflect"

type S struct {
    A int
    b string
}

func (s *S) String() { return s.b }

s := &S{
    A: 5,
    b: "example",
}

indirect := reflect.ValueOf(s) // effectively a pointer to an S
value := indirect.Elem()       // this is addressable, since we've derefed a pointer

value.FieldByName("A").Interface() // 5
value.Field(2).Interface()         // "example"

value.NumMethod()    // 0, since String takes a pointer receiver
indirect.NumMethod() // 1

indirect.Method(0).Call([]reflect.Value{})              // "example"
indirect.MethodByName("String").Call([]reflect.Value{}) // "example"
```

```
import "reflect"

s := []int{1, 2, 3}

value := reflect.ValueOf(s)

value.Len()                // 3
value.Index(0).Interface() // 1
value.Type().Kind()        // reflect.Slice
value.Type().Elem().Name() // int

value.Index(1).CanAddr()   // true -- slice elements are addressable
value.Index(1).CanSet()    // true -- and settable
value.Index(1).Set(5)

typ := reflect.SliceOf(reflect.TypeOf("example"))
newS := reflect.MakeSlice(typ, 0, 10) // an empty []string{} with capacity 10
```

reflect.Value.Elem

```
import "reflect"

// this is effectively a pointer dereference

x := 5
ptr := reflect.ValueOf(&x)
ptr.Type().Name() // *int
ptr.Type().Kind() // reflect.Ptr
ptr.Interface()   // [pointer to x]
ptr.Set(4)        // panic

value := ptr.Elem() // this is a deref
value.Type().Name() // int
value.Type().Kind() // reflect.Int
value.Set(4)        // this works
value.Interface()   // 4
```

- “”

reflect.TypeOf

```
package main

    import (
        "fmt"
        "reflect"
    )
    type Data struct {
     a int
    }
    func main() {
        s:="hey dude"
        fmt.Println(reflect.TypeOf(s))

        D := Data{a:5}
```

```
        fmt.Println(reflect.TypeOf(D))

    }
```

main.Data

https://riptutorial.com/zh-CN/go/topic/1854/

34: /

- func PlainAuthAuth
- func SendMailaddr stringAuthfrom stringto [] stringmsg [] byte

Examples

smtp.SendMail

Go。RFC 822RFC 822。

```
package main

import (
    "fmt"
    "net/smtp"
)

func main() {
    // user we are authorizing as
    from := "someuser@example.com"

    // use we are sending email to
    to := "otheruser@example.com"

    // server we are authorized to send email through
    host := "mail.example.com"

    // Create the authentication for the SendMail()
    // using PlainText, but other authentication methods are encouraged
    auth := smtp.PlainAuth("", from, "password", host)

    // NOTE: Using the backtick here ` works like a heredoc, which is why all the
    // rest of the lines are forced to the beginning of the line, otherwise the
    // formatting is wrong for the RFC 822 style
    message := `To: "Some User" <someuser@example.com>
From: "Other User" <otheruser@example.com>
Subject: Testing Email From Go!!

This is the message we are sending. That's it!
`

    if err := smtp.SendMail(host+":25", auth, from, []string{to}, []byte(message)); err != nil
{
        fmt.Println("Error SendMail: ", err)
        os.Exit(1)
    }
    fmt.Println("Email Sent!")
}
```

```
To: "Other User" <otheruser@example.com>
From: "Some User" <someuser@example.com>
Subject: Testing Email From Go!!
```

```
This is the message we are sending. That's it!
.
```

/ https://riptutorial.com/zh-CN/go/topic/5912/-

35:

- var x int //intx
- var s string //s
- x = 4 //x
- s =“foo”//s
- y= 5 //yint
- f= 4.5 //ffloat64
- b=“bar”//b

Examples

Go。

```
// Basic variable declaration. Declares a variable of type specified on the right.
// The variable is initialized to the zero value of the respective type.
var x int
var s string
var p Person // Assuming type Person struct {}

// Assignment of a value to a variable
x = 3

// Short declaration using := infers the type
y := 4

u := int64(100)    // declare variable of type int64 and init with 100
var u2 int64 = 100 // declare variable of type int64 and init with 100
```

Go。

```
// You can declare multiple variables of the same type in one line
var a, b, c string

var d, e string = "Hello", "world!"

// You can also use short declaration to assign multiple variables
x, y, z := 1, 2, 3

foo, bar := 4, "stack" // `foo` is type `int`, `bar` is type `string`
```

。

```
func multipleReturn() (int, int) {
    return 1, 2
}

x, y := multipleReturn() // x = 1, y = 2

func multipleReturn2() (a int, b int) {
    a = 3
```

```
    b = 4
    return
}

w, z := multipleReturn2() // w = 3, z = 4
```

Go。 。 “” _。

。

```
func SumProduct(a, b int) (int, int) {
    return a+b, a*b
}

func main() {
    // I only want the sum, but not the product
    sum, _ := SumProduct(1,2) // the product gets discarded
    fmt.Println(sum) // prints 3
}
```

range

```
func main() {

    pets := []string{"dog", "cat", "fish"}

    // Range returns both the current index and value
    // but sometimes you may only want to use the value
    for _, pet := range pets {
        fmt.Println(pet)
    }

}
```

。 var.(type)

```
x := someFunction() // Some value of an unknown type is stored in x now

switch x := x.(type) {
    case bool:
        fmt.Printf("boolean %t\n", x)             // x has type bool
    case int:
        fmt.Printf("integer %d\n", x)             // x has type int
    case string:
        fmt.Printf("pointer to boolean %s\n", x) // x has type string
    default:
        fmt.Printf("unexpected type %T\n", x)     // %T prints whatever type x is
}
```

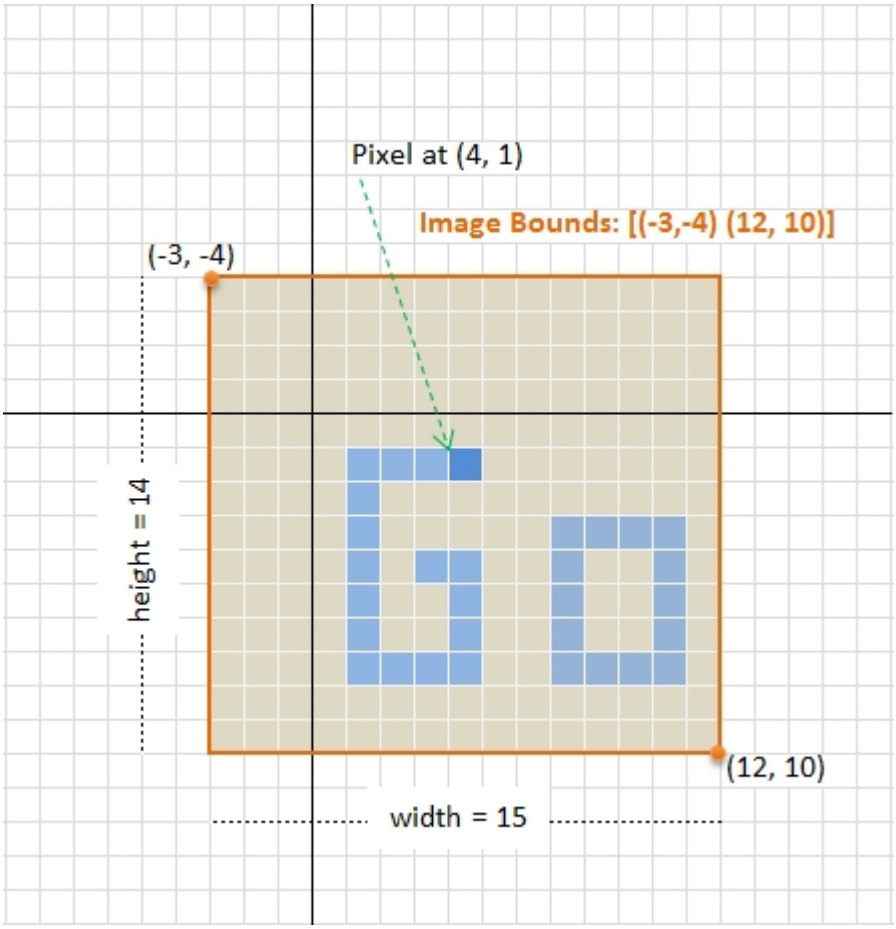
https://riptutorial.com/zh-CN/go/topic/674/

36:

◦ ◦

Examples

◦ / ◦ 2-dimage.Rectangle image.Point◦



◦ 15x14-3-4 1010◦ **0,00,0** ◦

Goimage.Image

```
type Image interface {
    // ColorModel returns the Image's color model.
    ColorModel() color.Model
    // Bounds returns the domain for which At can return non-zero color.
    // The bounds do not necessarily contain the point (0, 0).
    Bounds() Rectangle
    // At returns the color of the pixel at (x, y).
    // At(Bounds().Min.X, Bounds().Min.Y) returns the upper-left pixel of the grid.
    // At(Bounds().Max.X-1, Bounds().Max.Y-1) returns the lower-right one.
    At(x, y int) color.Color
}
```

color.Color

```
type Color interface {
    // RGBA returns the alpha-premultiplied red, green, blue and alpha values
    // for the color. Each value ranges within [0, 0xffff], but is represented
    // by a uint32 so that multiplying by a blend factor up to 0xffff will not
    // overflow.
    //
    // An alpha-premultiplied color component c has been scaled by alpha (a),
    // so has valid values 0 <= c <= a.
    RGBA() (r, g, b, a uint32)
}
```

color.Model

```
type Model interface {
    Convert(c Color) Color
}
```

img

```
// Image bounds and dimension
b := img.Bounds()
width, height := b.Dx(), b.Dy()
// do something with dimension ...

// Corner co-ordinates
top := b.Min.Y
left := b.Min.X
bottom := b.Max.Y
right := b.Max.X

// Accessing pixel. The (x,y) position must be
// started from (left, top) position not (0,0)
for y := top; y < bottom; y++ {
    for x := left; x < right; x++ {
        cl := img.At(x, y)
        r, g, b, a := cl.RGBA()
        // do something with r,g,b,a color component
    }
}
```

R,G,B,A0-65535 0x0000 - 0xffff **0-255** ◦

◦ RAW PNGJPEGGIF◦ image.Image◦ image.Decode

```
func Decode(r io.Reader) (Image, string, error)
```

◦ image.Decodeimage.RegisterFormat

```
func RegisterFormat(name, magic string,
    decode func(io.Reader) (Image, error), decodeConfig func(io.Reader) (Config, error))
```

JPEG GIFPNG ◦

```
import _ "image/jpeg"    //register JPEG decoder
```

main◦ mainJPEG

```
f, err := os.Open("inputimage.jpg")
if err != nil {
    // Handle error
}
defer f.Close()

img, fmtName, err := image.Decode(f)
if err != nil {
    // Handle error
}

// `fmtName` contains the name used during format registration
// Work with `img` ...
```

PNG

```
import "image/png"    //needed to use `png` encoder
```

```
f, err := os.Create("outimage.png")
if err != nil {
    // Handle error
}
defer f.Close()

// Encode to `PNG` with `DefaultCompression` level
// then save to file
err = png.Encode(f, img)
if err != nil {
    // Handle error
}
```

DefaultCompression

```
enc := png.Encoder{
    CompressionLevel: png.BestSpeed,
}
err := enc.Encode(f, img)
```

JPEG

jpeg

```
import "image/jpeg"

// Somewhere in the same package
f, err := os.Create("outimage.jpg")
if err != nil {
    // Handle error
}
defer f.Close()

// Specify the quality, between 0-100
```

```
// Higher is better
opt := jpeg.Options{
    Quality: 90,
}
err = jpeg.Encode(f, img, &opt)
if err != nil {
    // Handle error
}
```

GIF

GIF。

```
import "image/gif"

// Samewhere in the same package
f, err := os.Create("outimage.gif")
if err != nil {
    // Handle error
}
defer f.Close()

opt := gif.Options {
    NumColors: 256,
    // Add more parameters as needed
}

err = gif.Encode(f, img, &opt)
if err != nil {
    // Handle error
}
```

`SubImage(r Rectangle) Image` `image.Uniform`。

```
func CropImage(img image.Image, cropRect image.Rectangle) (cropImg image.Image, newImg bool) {
    //Interface for asserting whether `img`
    //implements SubImage or not.
    //This can be defined globally.
    type CropableImage interface {
        image.Image
        SubImage(r image.Rectangle) image.Image
    }

    if p, ok := img.(CropableImage); ok {
        // Call SubImage. This should be fast,
        // since SubImage (usually) shares underlying pixel.
        cropImg = p.SubImage(cropRect)
    } else if cropRect = cropRect.Intersect(img.Bounds()); !cropRect.Empty() {
        // If `img` does not implement `SubImage`,
        // copy (and silently convert) the image portion to RGBA image.
        rgbaImg := image.NewRGBA(cropRect)
        for y := cropRect.Min.Y; y < cropRect.Max.Y; y++ {
            for x := cropRect.Min.X; x < cropRect.Max.X; x++ {
                rgbaImg.Set(x, y, img.At(x, y))
            }
        }
        cropImg = rgbaImg
```

```
        newImg = true
    } else {
        // Return an empty RGBA image
        cropImg = &image.RGBA{}
        newImg = true
    }

    return cropImg, newImg
}
```

◦ ◦

◦ R, G, B◦ ◦

8◦ net/httpPNG◦

```
package main

import (
    "image"
    "log"
    "net/http"
    "os"

    _ "image/jpeg"
    "image/png"
)

func main() {
    // Load image from remote through http
    // The Go gopher was designed by Renee French. (http://reneefrench.blogspot.com/)
    // Images are available under the Creative Commons 3.0 Attributions license.
    resp, err := http.Get("http://golang.org/doc/gopher/fiveyears.jpg")
    if err != nil {
        // handle error
        log.Fatal(err)
    }
    defer resp.Body.Close()

    // Decode image to JPEG
    img, _, err := image.Decode(resp.Body)
    if err != nil {
        // handle error
        log.Fatal(err)
    }
    log.Printf("Image type: %T", img)

    // Converting image to grayscale
    grayImg := image.NewGray(img.Bounds())
    for y := img.Bounds().Min.Y; y < img.Bounds().Max.Y; y++ {
        for x := img.Bounds().Min.X; x < img.Bounds().Max.X; x++ {
            grayImg.Set(x, y, img.At(x, y))
        }
    }

    // Working with grayscale image, e.g. convert to png
    f, err := os.Create("fiveyears_gray.png")
    if err != nil {
        // handle error
```

```
        log.Fatal(err)
    }
    defer f.Close()

    if err := png.Encode(f, grayImg); err != nil {
        log.Fatal(err)
    }
}
```

image.goSet(x, y int, c color.Color)

```
func (p *Gray) Set(x, y int, c color.Color) {
    if !(Point{x, y}.In(p.Rect)) {
        return
    }

    i := p.PixOffset(x, y)
    p.Pix[i] = color.GrayModel.Convert(c).(color.Gray).Y
}
```

color.GrayModelcolor.go

```
func grayModel(c Color) Color {
    if _, ok := c.(Gray); ok {
        return c
    }
    r, g, b, _ := c.RGBA()

    // These coefficients (the fractions 0.299, 0.587 and 0.114) are the same
    // as those given by the JFIF specification and used by func RGBToYCbCr in
    // ycbcr.go.
    //
    // Note that 19595 + 38470 + 7471 equals 65536.
    //
    // The 24 is 16 + 8. The 16 is the same as used in RGBToYCbCr. The 8 is
    // because the return value is 8 bit color, not 16 bit color.
    y := (19595*r + 38470*g + 7471*b + 1<<15) >> 24

    return Gray{uint8(y)}
}
```

Y

Y = 0.299 **R** + 0.587 **G** + 0.114 **B.**

/

Y = **R** + **G** + **B** / 3
LumaY = 0.2126 **R** + 0.7152 **G** + 0.0722 **B.**
Y =min **R** **G** **B** + max **R** **G** **B** / 2

。

```
// Converting image to grayscale
grayImg := image.NewGray(img.Bounds())
for y := img.Bounds().Min.Y; y < img.Bounds().Max.Y; y++ {
```

```
    for x := img.Bounds().Min.X; x < img.Bounds().Max.X; x++ {
        R, G, B, _ := img.At(x, y).RGBA()
        //Luma: Y = 0.2126*R + 0.7152*G + 0.0722*B
        Y := (0.2126*float64(R) + 0.7152*float64(G) + 0.0722*float64(B)) * (255.0 / 65535)
        grayPix := color.Gray{uint8(Y)}
        grayImg.Set(x, y, grayPix)
    }
}
```

◦ color.GraySet(x, y int, c color.Color)grayModel◦

https://riptutorial.com/zh-CN/go/topic/10557/

37:

。 。 。

- var mapName map [KeyType] ValueType //Map
- var mapName = map [KeyType] ValueType {} //Map
- var mapName = map [KeyType] ValueType {key1val1key2val2} //Map
- mapName= makemap [KeyType] ValueType//
- mapName= makemap [KeyType] ValueTypelength//
- mapName= map [KeyType] ValueType {} //=Map
- mapName= map [KeyType] ValueType {key1value1key2value2} //Map=
- value= mapName [key] //
- valuehasKey= mapName [key] //'hasKey''true'
- mapName [key] = value //

Go`map`。 Go 。

Examples

`mapmap`

```
// Keys are ints, values are ints.
var m1 map[int]int // initialized to nil

// Keys are strings, values are ints.
var m2 map[string]int // initialized to nil
```

`nil` 。 nil 。

`make`

```
m := make(map[string]int)
```

`make`

```
m := make(map[string]int, 30)
```

json。

```
m := make(map[string]int, 0)
```

`{}` 。

```
var m map[string]int = map[string]int{"Foo": 20, "Bar": 30}

fmt.Println(m["Foo"]) // outputs 20
```

。

```
// Declare, initializing to zero value, then assign a literal value.
var m map[string]int
m = map[string]int{}

// Declare and initialize via literal value.
var m = map[string]int{}

// Declare via short variable declaration and initialize with a literal value.
m := map[string]int{}
```

/ 。

; 。 interface{} 。

```
type Person struct {
    FirstName string
    LastName  string
}

// Declare via short variable declaration and initialize with make.
m := make(map[string]Person)

// Declare, initializing to zero value, then assign a literal value.
var m map[string]Person
m = map[string]Person{}

// Declare and initialize via literal value.
var m = map[string]Person{}

// Declare via short variable declaration and initialize with a literal value.
m := map[string]Person{}
```

。

```
mapIntInt := map[int]int{10: 100, 20: 100, 30: 1000}
mapIntString := map[int]string{10: "foo", 20: "bar", 30: "baz"}
mapStringInt := map[string]int{"foo": 10, "bar": 20, "baz": 30}
mapStringString := map[string]string{"foo": "one", "bar": "two", "baz": "three"}
```

Variable

```
var mapIntInt = map[int]int{10: 100, 20: 100, 30: 1000}
var mapIntString = map[int]string{10: "foo", 20: "bar", 30: "baz"}
var mapStringInt = map[string]int{"foo": 10, "bar": 20, "baz": 30}
var mapStringString = map[string]string{"foo": "one", "bar": "two", "baz": "three"}
```

```
// Custom struct types
type Person struct {
  FirstName, LastName string
}

var mapStringPerson = map[string]Person{
  "john": Person{"John", "Doe"},
```

```
  "jane": Person{"Jane", "Doe"}}
mapStringPerson := map[string]Person{
  "john": Person{"John", "Doe"},
  "jane": Person{"Jane", "Doe"}}
```

```
type RouteHit struct {
    Domain string
    Route  string
}

var hitMap = map[RouteHit]int{
  RouteHit{"example.com","/home"}: 1,
  RouteHit{"example.com","/help"}: 2}
hitMap := map[RouteHit]int{
  RouteHit{"example.com","/home"}: 1,
  RouteHit{"example.com","/help"}: 2}
```

{}◦

```
mapIntInt := map[int]int{}
mapIntString := map[int]string{}
mapStringInt := map[string]int{}
mapStringString := map[string]string{}
mapStringPerson := map[string]Person{}
```

◦ ◦

```
// using a map as argument for fmt.Println()
fmt.Println(map[string]string{
  "FirstName": "John",
  "LastName": "Doe",
  "Age": "30"})

// equivalent to
data := map[string]string{
  "FirstName": "John",
  "LastName": "Doe",
  "Age": "30"}
fmt.Println(data)
```

mapnil 0 ◦

```
var m map[string]string
fmt.Println(m == nil) // true
fmt.Println(len(m) ==0) // true
```

nil◦ nil◦

```
var m map[string]string

// reading
m["foo"] == "" // true. Remember "" is the zero value for a string
_, ok = m["foo"] // ok == false
```

```
// writing
m["foo"] = "bar" // panic: assignment to entry in nil map
```

◦ make◦

```
var m map[string]string
m = make(map[string]string) // OR m = map[string]string{}
m["foo"] = "bar"
```

```
import fmt

people := map[string]int{
  "john": 30,
  "jane": 29,
  "mark": 11,
}

for key, value := range people {
  fmt.Println("Name:", key, "Age:", value)
}
```

◦

◦

```
people := map[string]int{
  "john": 30,
  "jane": 29,
  "mark": 11,
}

for key, _ := range people {
  fmt.Println("Name:", key)
}
```

```
for key := range people {
  fmt.Println("Name:", key)
}
```

◦

delete◦

```
people := map[string]int{"john": 30, "jane": 29}
fmt.Println(people) // map[john:30 jane:29]

delete(people, "john")
fmt.Println(people) // map[jane:29]
```

mapnildelete◦

```
people := map[string]int{"john": 30, "jane": 29}
```

```
fmt.Println(people) // map[john:30 jane:29]

delete(people, "notfound")
fmt.Println(people) // map[john:30 jane:29]

var something map[string]int
delete(something, "notfound") // no-op
```

lenmap

```
m := map[string]int{}
len(m) // 0

m["foo"] = 1
len(m) // 1
```

nillen0。

```
var m map[string]int
len(m) // 0
```

go。 。 sync.RWMutex。 sync.Mutex 。

```
type RWMap struct {
    sync.RWMutex
    m map[string]int
}

// Get is a wrapper for getting the value from the underlying map
func (r RWMap) Get(key string) int {
    r.RLock()
    defer r.RUnlock()
    return r.m[key]
}

// Set is a wrapper for setting the value of a key in the underlying map
func (r RWMap) Set(key string, val int) {
    r.Lock()
    defer r.Unlock()
    r.m[key] = val
}

// Inc increases the value in the RWMap for a key.
//   This is more pleasant than r.Set(key, r.Get(key)++)
func (r RWMap) Inc(key string) {
    r.Lock()
    defer r.Unlock()
    r.m[key]++
}

func main() {

    // Init
    counter := RWMap{m: make(map[string]int)}

    // Get a Read Lock
    counter.RLock()
```

```
    _ = counter.["Key"]
    counter.RUnlock()

    // the above could be replaced with
    _ = counter.Get("Key")

    // Get a write Lock
    counter.Lock()
    counter.m["some_key"]++
    counter.Unlock()

    // above would need to be written as
    counter.Inc("some_key")
}
```

。

```
m := make(map[string][]int)
```

nil。 nilappend

```
// m["key1"] == nil && len(m["key1"]) == 0
m["key1"] = append(m["key1"], 1)
// len(m["key1"]) == 1
```

```
delete(m, "key1")
// m["key1"] == nil
```

00

```
value := mapName[ key ]
```

。

0 int ""string...

```
m  := map[string]string{"foo": "foo_value", "bar": ""}
k  := m["foo"]  // returns "foo_value" since that is the value stored in the map
k2 := m["bar"]  // returns "" since that is the value stored in the map
k3 := m["nop"]  // returns "" since the key does not exist, and "" is the string type's zero
value
```

value, hasKey := map["key"] 。

boolean

- true
- false 。

```
value, hasKey = m[ key ]
if hasKey {
    // the map contains the given key, so we can safely use the value
    // If value is zero-value, it's because the zero-value was pushed to the map
```

```
} else {
    // The map does not have the given key
    // the value will be the zero-value of the map's type
}
```

```
people := map[string]int{
  "john": 30,
  "jane": 29,
  "mark": 11,
}

for _, value := range people {
  fmt.Println("Age:", value)
}
```

◦

◦ ◦

```
// Create the original map
originalMap := make(map[string]int)
originalMap["one"] = 1
originalMap["two"] = 2

// Create the target map
targetMap := make(map[string]int)

// Copy from the original map to the target map
for key, value := range originalMap {
  targetMap[key] = value
}
```

◦ Go `map[Type]struct{}` `map[Type]struct{}` ◦

```
// To initialize a set of strings:
greetings := map[string]struct{}{
    "hi":    {},
    "hello": {},
}

// To delete a value:
delete(greetings, "hi")

// To add a value:
greetings["hey"] = struct{}{}

// To check if a value is in the set:
if _, ok := greetings["hey"]; ok {
    fmt.Println("hey is in greetings")
}
```

https://riptutorial.com/zh-CN/go/topic/732/

38:

goroutineGo。

fxyz

goroutine

fxyz

fxyzgoroutinefgoroutine。

Goroutines。 Go。

https //tour.golang.org/concurrency/1

Examples

Goroutines

```
package main

import (
    "fmt"
    "time"
)

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}

func main() {
    go say("world")
    say("hello")
}
```

goroutine。 goroutinego

```
package main

import "fmt"

func f(n int) {
  for i := 0; i < 10; i++ {
    fmt.Println(n, ":", i)
  }
}

func main() {
```

```
  go f(0)
  var input string
  fmt.Scanln(&input)
}
```

◦ goroutines◦ `Scanln`◦

https://riptutorial.com/zh-CN/go/topic/9776/

39:

Examples

LinuxUbuntu

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential git curl wget
$ wget https://storage.googleapis.com/golang/go<versions>.gz
```

。

```
# To install go1.7 use
$ wget https://storage.googleapis.com/golang/go1.7.linux-amd64.tar.gz


# Untar the file
$ sudo tar -C /usr/local -xzf go1.7.linux-amd64.tar.gz
$ sudo chown -R $USER:$USER /usr/local/go
$ rm go1.5.4.linux-amd64.tar.gz
```

$GOPATH

```
$ mkdir $HOME/go
```

/ .bashrc

```
export GOPATH=$HOME/go
export PATH=$GOPATH/bin:/usr/local/go/bin:$PATH
```

```
$ nano ~/.bashrc
  export GOPATH=$HOME/go
  export PATH=$GOPATH/bin:/usr/local/go/bin:$PATH

$ source ~/.bashrc
```

go

```
$ go version
go version go<version> linux/amd64
```

https://riptutorial.com/zh-CN/go/topic/5776/

40:

Go

。 。

go [version]。[operating system] - [architecture]。[archive]

。 。

Mac“darwin”。Mac。

64“”“amd64”。32“386”。Raspberry PiARM“armv6l”。

“”MacWindowsGo。Mac“pkg”。Windows“msi”。

64WindowsGo 1.6.3

```
go1.6.3.windows-amd64.msi
```

Go。

MacWindows

。 。

Linux

Linux。“.tar.gz”。“.zip”。。Go`/usr/local`。

```
。Downloads。。
```

```
cd Downloads
```

```
/usr/local [filename]。
```

```
tar -C /usr/local -xzf [filename].tar.gz
```

。 。

```
GOPATH。
```

“”“”“”。Windows“”“”。

“GOPATH”。Go code in。“go”。

`GOPATH`。

“.bash_profile”`[work area]`。 “.bash_profile”。“go”。

```
export GOPATH=[work area]
```

Linux

Linux。 Go`GOPATH`。

“.profile”`[work area]`tto。“ .profile“。“go”。

“.profile”。

```
export PATH=$PATH:/usr/local/go/bin
```

Go。

Examples

.profile.bash_profile

```
# This is an example of a .profile or .bash_profile for Linux and Mac systems
export GOPATH=/home/user/go
export PATH=$PATH:/usr/local/go/bin
```

https://riptutorial.com/zh-CN/go/topic/6213/

41:

Examples

```
package main

import (
    "fmt"
    "sync"
)

type job struct {
    // some fields for your job type
}

type result struct {
    // some fields for your result type
}

func worker(jobs <-chan job, results chan<- result) {
    for j := range jobs {
        var r result
        // do some work
        results <- r
    }
}

func main() {
    // make our channels for communicating work and results
    jobs := make(chan job, 100) // 100 was chosen arbitrarily
    results := make(chan result, 100)

    // spin up workers and use a sync.WaitGroup to indicate completion
    wg := sync.WaitGroup
    for i := 0; i < runtime.NumCPU; i++ {
        wg.Add(1)
        go func() {
            defer wg.Done()
            worker(jobs, results)
        }()
    }

    // wait on the workers to finish and close the result channel
    // to signal downstream that all work is done
    go func() {
        defer close(results)
        wg.Wait()
    }()

    // start sending jobs
    go func() {
        defer close(jobs)
        for {
            jobs <- getJob()    // I haven't defined getJob() and noMoreJobs()
            if noMoreJobs() {  // they are just for illustration
                break
            }
        }
```

```
    }()

    // read all the results
    for r := range results {
        fmt.Println(r)
    }
}
```

Web

```
package main

import (
  "fmt"
  "runtime"
  "strconv"
  "sync"
  "time"
)

// Job - interface for job processing
type Job interface {
  Process()
}

// Worker - the worker threads that actually process the jobs
type Worker struct {
  done             sync.WaitGroup
  readyPool        chan chan Job
  assignedJobQueue chan Job

  quit chan bool
}

// JobQueue - a queue for enqueueing jobs to be processed
type JobQueue struct {
  internalQueue     chan Job
  readyPool         chan chan Job
  workers           []*Worker
  dispatcherStopped sync.WaitGroup
  workersStopped    sync.WaitGroup
  quit              chan bool
}

// NewJobQueue - creates a new job queue
func NewJobQueue(maxWorkers int) *JobQueue {
  workersStopped := sync.WaitGroup{}
  readyPool := make(chan chan Job, maxWorkers)
  workers := make([]*Worker, maxWorkers, maxWorkers)
  for i := 0; i < maxWorkers; i++ {
    workers[i] = NewWorker(readyPool, workersStopped)
  }
  return &JobQueue{
    internalQueue:     make(chan Job),
    readyPool:         readyPool,
    workers:           workers,
    dispatcherStopped: sync.WaitGroup{},
    workersStopped:    workersStopped,
    quit:              make(chan bool),
  }
```

```
}

// Start - starts the worker routines and dispatcher routine
func (q *JobQueue) Start() {
  for i := 0; i < len(q.workers); i++ {
    q.workers[i].Start()
  }
  go q.dispatch()
}

// Stop - stops the workers and sispatcher routine
func (q *JobQueue) Stop() {
  q.quit <- true
  q.dispatcherStopped.Wait()
}

func (q *JobQueue) dispatch() {
  q.dispatcherStopped.Add(1)
  for {
    select {
    case job := <-q.internalQueue: // We got something in on our queue
      workerChannel := <-q.readyPool // Check out an available worker
      workerChannel <- job           // Send the request to the channel
    case <-q.quit:
      for i := 0; i < len(q.workers); i++ {
        q.workers[i].Stop()
      }
      q.workersStopped.Wait()
      q.dispatcherStopped.Done()
      return
    }
  }
}

// Submit - adds a new job to be processed
func (q *JobQueue) Submit(job Job) {
  q.internalQueue <- job
}

// NewWorker - creates a new worker
func NewWorker(readyPool chan chan Job, done sync.WaitGroup) *Worker {
  return &Worker{
    done:             done,
    readyPool:        readyPool,
    assignedJobQueue: make(chan Job),
    quit:             make(chan bool),
  }
}

// Start - begins the job processing loop for the worker
func (w *Worker) Start() {
  go func() {
    w.done.Add(1)
    for {
      w.readyPool <- w.assignedJobQueue // check the job queue in
      select {
      case job := <-w.assignedJobQueue: // see if anything has been assigned to the queue
        job.Process()
      case <-w.quit:
        w.done.Done()
        return
```

```
      }
    }
  }()
}

// Stop - stops the worker
func (w *Worker) Stop() {
  w.quit <- true
}

//////////////// Example /////////////////

// TestJob - holds only an ID to show state
type TestJob struct {
  ID string
}

// Process - test process function
func (t *TestJob) Process() {
  fmt.Printf("Processing job '%s'\n", t.ID)
  time.Sleep(1 * time.Second)
}

func main() {
  queue := NewJobQueue(runtime.NumCPU())
  queue.Start()
  defer queue.Stop()

  for i := 0; i < 4*runtime.NumCPU(); i++ {
    queue.Submit(&TestJob{strconv.Itoa(i)})
  }
}
```

https://riptutorial.com/zh-CN/go/topic/4182/

42:

Go。

Examples

const

```
const Greeting string = "Hello World"
const Years int = 10
const Truth bool = true
```

。

```
// not exported
const alpha string = "Alpha"
// exported
const Beta string = "Beta"
```

。

```
package main

import (
    "fmt"
    "math"
)

const s string = "constant"

func main() {
    fmt.Println(s) // constant

    // A `const` statement can appear anywhere a `var` statement can.
    const n = 10
    fmt.Println(n)                              // 10
    fmt.Printf("n=%d is of type %T\n", n, n) // n=10 is of type int

    const m float64 = 4.3
    fmt.Println(m) // 4.3

    // An untyped constant takes the type needed by its context.
    // For example, here `math.Sin` expects a `float64`.
    const x = 10
    fmt.Println(math.Sin(x)) // -0.5440211108893699
}
```

const

```
const (
    Alpha = "alpha"
    Beta  = "beta"
    Gamma = "gamma"
```

```
)
```

`iota`

```
const (
    Zero = iota // Zero == 0
    One         // One  == 1
    Two         // Two  == 2
)
```

`iota`Iota 。

。 。

```
const Foo, Bar = "foo", "bar"
```

Go。

```
"bar"
```

`string` 。 *Untyped* 。 `string` Go 。 。

```
const foo = "bar"
```

。

```
const typedFoo string = "bar"
```

。

```
var s string
s = foo      // This works just fine
s = typedFoo // As does this

type MyString string
var mys MyString
mys = foo      // This works just fine
mys = typedFoo // cannot use typedFoo (type string) as type MyString in assignment
```

https://riptutorial.com/zh-CN/go/topic/1047/

43:

Gogoroutinegoroutine。 。

- go doWork//doWorkgoroutine
- ch= makechan int//int
- ch < - 1 //
- value = <-ch //

GoGoroutines。 GoGOMAXPROCSgoroutines。 Go。

Examples

goroutines

gogoroutine

```
func DoMultiply(x,y int) {
    // Simulate some hard work
    time.Sleep(time.Second * 1)
    fmt.Printf("Result: %d\n", x * y)
}

go DoMultiply(1,2) // first execution, non-blocking
go DoMultiply(3,4) // second execution, also non-blocking

// Results are printed after a single second only,
// not 2 seconds because they execute concurrently:
// Result: 2
// Result: 12
```

。

Hello World Goroutine

goroutine。

```
package main

import "fmt"
import "time"

func main() {
    // create new channel of type string
    ch := make(chan string)

    // start new anonymous goroutine
    go func() {
        time.Sleep(time.Second)
        // send "Hello World" to channel
        ch <- "Hello World"
```

```
    }()
    // read from channel
    msg, ok := <-ch
    fmt.Printf("msg='%s', ok='%v'\n", msg, ok)
}
```

ch 。

time.Sleepmain()ch goroutine <-chmain() 。 main()goroutine。

goroutines

main goroutine。 sync.WaitGroup。

```
package main

import (
    "fmt"
    "sync"
)

var wg sync.WaitGroup // 1

func routine(i int) {
    defer wg.Done() // 3
    fmt.Printf("routine %v finished\n", i)
}

func main() {
    wg.Add(10) // 2
    for i := 0; i < 10; i++ {
        go routine(i) // *
    }
    wg.Wait() // 4
    fmt.Println("main finished")
}
```

WaitGroup

1. 。 。
2. 。 goroutinegoroutine4。
3. 。 goroutine。 。
4. 0.goroutinegoroutine。

* goroutine。 wg.Add(10)。 WaitGroup10wg.Wait10wg.Waitmain() goroutine。 ifor。

goroutine

val。 valgoroutine

```
for val := range values {
    go func(val interface{}) {
        fmt.Println(val)
```

```
    }(val)
}
```

`func(val interface{}) { ... }()`VAL`val`val。

```
for val := range values {
    val := val
    go func() {
        fmt.Println(val)
    }()
}
```

`val := val`goroutine。

goroutines

```
package main

import (
    "log"
    "sync"
    "time"
)

func main() {
    // The WaitGroup lets the main goroutine wait for all other goroutines
    // to terminate. However, this is no implicit in Go. The WaitGroup must
    // be explicitely incremented prior to the execution of any goroutine
    // (i.e. before the `go` keyword) and it must be decremented by calling
    // wg.Done() at the end of every goroutine (typically via the `defer` keyword).
    wg := sync.WaitGroup{}

    // The stop channel is an unbuffered channel that is closed when the main
    // thread wants all other goroutines to terminate (there is no way to
    // interrupt another goroutine in Go). Each goroutine must multiplex its
    // work with the stop channel to guarantee liveness.
    stopCh := make(chan struct{})


    for i := 0; i < 5; i++ {
        // It is important that the WaitGroup is incremented before we start
        // the goroutine (and not within the goroutine) because the scheduler
        // makes no guarantee that the goroutine starts execution prior to
        // the main goroutine calling wg.Wait().
        wg.Add(1)
        go func(i int, stopCh <-chan struct{}) {
            // The defer keyword guarantees that the WaitGroup count is
            // decremented when the goroutine exits.
            defer wg.Done()

            log.Printf("started goroutine %d", i)

            select {
            // Since we never send empty structs on this channel we can
            // take the return of a receive on the channel to mean that the
            // channel has been closed (recall that receive never blocks on
            // closed channels).
```

```
            case <-stopCh:
                log.Printf("stopped goroutine %d", i)
            }
        }(i, stopCh)
    }

    time.Sleep(time.Second * 5)
    close(stopCh)
    log.Printf("stopping goroutines")
    wg.Wait()
    log.Printf("all goroutines stopped")
}
```

goroutines

```
package main

import (
    "fmt"
    "time"
)

// The pinger prints a ping and waits for a pong
func pinger(pinger <-chan int, ponger chan<- int) {
    for {
        <-pinger
        fmt.Println("ping")
        time.Sleep(time.Second)
        ponger <- 1
    }
}

// The ponger prints a pong and waits for a ping
func ponger(pinger chan<- int, ponger <-chan int) {
    for {
        <-ponger
        fmt.Println("pong")
        time.Sleep(time.Second)
        pinger <- 1
    }
}

func main() {
    ping := make(chan int)
    pong := make(chan int)

    go pinger(ping, pong)
    go ponger(ping, pong)

    // The main goroutine starts the ping/pong by sending into the ping channel
    ping <- 1

    for {
        // Block the main thread until an interrupt
        time.Sleep(time.Second)
    }
}
```

Go Playground

https://riptutorial.com/zh-CN/go/topic/376/

44:

- // +

。 。

Go。Gopackage。

。

Examples

。

```
// +build integration

package main

import (
    "testing"
)

func TestThatRequiresNetworkAccess(t *testing.T) {
    t.Fatal("It failed!")
}
```

`go test`

```
go test -tags "integration"
```

```
$ go test
?       bitbucket.org/yourname/yourproject    [no test files]
$ go test -tags "integration"
--- FAIL: TestThatRequiresNetworkAccess (0.00s)
        main_test.go:10: It failed!
FAIL
exit status 1
FAIL    bitbucket.org/yourname/yourproject    0.003s
```

/xorxor`crypto/cipher/xor.gocrypto/cipher/xor.go`

```
// +build 386 amd64 s390x

package cipher

func xorBytes(dst, a, b []byte) int { /* This function uses unaligned reads / writes to
optimize the operation */ }
```

```
// +build !386,!amd64,!s390x

package cipher
```

```
func xorBytes(dst, a, b []byte) int { /* This version of the function just loops and xors */ }
```

https://riptutorial.com/zh-CN/go/topic/2595/

45:

◦ ◦ Go◦

Examples

`for`go

```
// like if, for doesn't use parens either.
// variables declared in for and if are local to their scope.
for x := 0; x < 3; x++ { // ++ is a statement.
    fmt.Println("iteration", x)
}

// would print:
// iteration 0
// iteration 1
// iteration 2
```

Go

```
for x := 0; x < 10; x++ { // loop through 0 to 9
    if x < 3 { // skips all the numbers before 3
        continue
    }
    if x > 5 { // breaks out of the loop once x == 6
        break
    }
    fmt.Println("iteration", x)
}

// would print:
// iteration 3
// iteration 4
// iteration 5
```

`break`continue

```
OuterLoop:
for {
    for {
        if allDone() {
            break OuterLoop
        }
        if innerDone() {
            continue OuterLoop
        }
        // do something
    }
}
```

`for`while◦

```
package main

import (
    "fmt"
)

func main() {
    i := 0
    for i < 3 { // Will repeat if condition is true
        i++
        fmt.Println(i)
    }
}
```

```
1
2
3
```

```
for {
    // This will run until a return or break.
}
```

```
for i := 0; i < 10; i++ {
    fmt.Print(i, " ")
}
```

```
for i, j := 0, 0; i < 5 && j < 10; i, j = i+1, j+2 {
    fmt.Println(i, j)
}
```

```
i := 0
for ; i < 10; i++ {
    fmt.Print(i, " ")
}
```

```
for i := 1; ; i++ {
    if i&1 == 1 {
        continue
    }
    if i == 22 {
        break
    }
    fmt.Print(i, " ")
}
```

```
for i := 0; i < 10; {
    fmt.Print(i, " ")
    i++
}
```

```
i := 0
for {
    fmt.Print(i, " ")
```

```
    i++
    if i == 10 {
        break
    }
}
```

```
for i := 0; ; {
    fmt.Print(i, " ")
    if i == 9 {
        break
    }
    i++
}
```

while

```
i := 0
for i < 10 {
    fmt.Print(i, " ")
    i++
}
```

```
i := 0
for ; ; i++ {
    fmt.Print(i, " ")
    if i == 9 {
        break
    }
}
```

```
ary := [5]int{0, 1, 2, 3, 4}
for index, value := range ary {
    fmt.Println("ary[", index, "] =", value)
}
```

```
for index := range ary {
    fmt.Println("ary[", index, "] =", ary[index])
}
```

```
for index, _ := range ary {
    fmt.Println("ary[", index, "] =", ary[index])
}
```

```
for _, value := range ary {
    fmt.Print(value, " ")
}
```

```
mp := map[string]int{"One": 1, "Two": 2, "Three": 3}
for key, value := range mp {
    fmt.Println("map[", key, "] =", value)
}
```

```
for key := range mp {
    fmt.Print(key, " ") //One Two Three
}
```

```
for key, _ := range mp {
    fmt.Print(key, " ") //One Two Three
}
```

map

```
for _, value := range mp {
    fmt.Print(value, " ") //2 3 1
}
```

```
ch := make(chan int, 10)
for i := 0; i < 10; i++ {
    ch <- i
}
close(ch)

for i := range ch {
    fmt.Print(i, " ")
}
```

Unicode

```
utf8str := "B = \u00b5H" //B = µH
for _, r := range utf8str {
    fmt.Print(r, " ") //66 32 61 32 181 72
}
fmt.Println()
for _, v := range []byte(utf8str) {
    fmt.Print(v, " ") //66 32 61 32 194 181 72
}
fmt.Println(len(utf8str)) //7
```

`utf8str`6Unicode7。

```
package main

import(
    "fmt"
    "time"
)

func main() {
    for _ = range time.Tick(time.Second * 3) {
        fmt.Println("Ticking every 3 seconds")
    }
}
```

https://riptutorial.com/zh-CN/go/topic/975/

46:

Defer Basics

Examples

◦ ◦ recover()main()◦ panicstderr ◦

```
package main

import "fmt"

func foo() {
    defer fmt.Println("Exiting foo")
    panic("bar")
}

func main() {
    defer fmt.Println("Exiting main")
    foo()
}
```

```
Exiting foo
Exiting main
panic: bar


goroutine 1 [running]:
panic(0x128360, 0x1040a130)
    /usr/local/go/src/runtime/panic.go:481 +0x700
main.foo()
    /tmp/sandbox550159908/main.go:7 +0x160
main.main()
    /tmp/sandbox550159908/main.go:12 +0x120
```

panic◦

panic◦ ◦ recover◦ ◦ recover()◦ recover()nil ◦

```
package main

import "fmt"

func foo() {
    panic("bar")
}

func bar() {
    defer func() {
        if msg := recover(); msg != nil {
            fmt.Printf("Recovered with message %s\n", msg)
        }
    }()
```

```
    foo()
    fmt.Println("Never gets executed")
}

func main() {
    fmt.Println("Entering main")
    bar()
    fmt.Println("Exiting main the normal way")
}
```

```
Entering main
Recovered with message bar
Exiting main the normal way
```

https://riptutorial.com/zh-CN/go/topic/4350/

47:

Examples

```
c := exec.Command(name, arg...)
b := &bytes.Buffer{}
c.Stdout = b
c.Stdin = stdin
if err := c.Start(); err != nil {
    return nil, err
}
timedOut := false
intTimer := time.AfterFunc(timeout, func() {
    log.Printf("Process taking too long. Interrupting: %s %s", name, strings.Join(arg, " "))
    c.Process.Signal(os.Interrupt)
    timedOut = true
})
killTimer := time.AfterFunc(timeout*2, func() {
    log.Printf("Process taking too long. Killing: %s %s", name, strings.Join(arg, " "))
    c.Process.Signal(os.Kill)
    timedOut = true
})
err := c.Wait()
intTimer.Stop()
killTimer.Stop()
if timedOut {
    log.Print("the process timed out\n")
}
```

```
// Execute a command a capture standard out. exec.Command creates the command
// and then the chained Output method gets standard out. Use CombinedOutput()
// if you want both standard out and standerr output
out, err := exec.Command("echo", "foo").Output()
if err != nil {
    log.Fatal(err)
}
```

```
cmd := exec.Command("sleep", "5")

// Does not wait for command to complete before returning
err := cmd.Start()
if err != nil {
    log.Fatal(err)
}

// Wait for cmd to Return
err = cmd.Wait()
log.Printf("Command finished with error: %v", err)
```

RunOutputCombinedOutputCmd

```
cmd := exec.Command("xte", "key XF86AudioPlay")
_ := cmd.Run() // Play audio key press
// .. do something else
```

```
err := cmd.Run() // Pause audio key press, fails
```

exec

exec.Command ◦ ◦

```
cmd := exec.Command("xte", "key XF86AudioPlay")
_ := cmd.Run() // Play audio key press
// .. wait a moment
cmd := exec.Command("xte", "key XF86AudioPlay")
_ := cmd.Run() // Pause audio key press
```

https://riptutorial.com/zh-CN/go/topic/1097/

48:

- pointer=variable //
- = *//
- * pointer = value //
- pointer= newStruct//

Examples

Go 。

```
package main

import "fmt"

// We'll show how pointers work in contrast to values with
// 2 functions: `zeroval` and `zeroptr`. `zeroval` has an
// `int` parameter, so arguments will be passed to it by
// value. `zeroval` will get a copy of `ival` distinct
// from the one in the calling function.
func zeroval(ival int) {
    ival = 0
}

// `zeroptr` in contrast has an `*int` parameter, meaning
// that it takes an `int` pointer. The `*iptr` code in the
// function body then _dereferences_ the pointer from its
// memory address to the current value at that address.
// Assigning a value to a dereferenced pointer changes the
// value at the referenced address.
func zeroptr(iptr *int) {
    *iptr = 0
}
```

```
func main() {
    i := 1
    fmt.Println("initial:", i) // initial: 1

    zeroval(i)
    fmt.Println("zeroval:", i) // zeroval: 1
    // `i` is still equal to 1 because `zeroval` edited
    // a "copy" of `i`, not the original.

    // The `&i` syntax gives the memory address of `i`,
    // i.e. a pointer to `i`. When calling `zeroptr`,
    // it will edit the "original" `i`.
    zeroptr(&i)
    fmt.Println("zeroptr:", i) // zeroptr: 0

    // Pointers can be printed too.
    fmt.Println("pointer:", &i) // pointer: 0x10434114
}
```

◦

Go Spec

◦ ◦ ◦ t.Mp(&t).Mp ◦

```
package main

import "fmt"

type Foo struct {
    Bar int
}

func (f *Foo) Increment() {
    f.Bar += 1
}

func main() {
    var f Foo

    // Calling `f.Increment` is automatically changed to `(&f).Increment` by the compiler.
    f = Foo{}
    fmt.Printf("f.Bar is %d\n", f.Bar)
    f.Increment()
    fmt.Printf("f.Bar is %d\n", f.Bar)

    // As you can see, calling `(&f).Increment` directly does the same thing.
    f = Foo{}
    fmt.Printf("f.Bar is %d\n", f.Bar)
    (&f).Increment()
    fmt.Printf("f.Bar is %d\n", f.Bar)
}
```

◦

Go Spec

◦ ◦ ◦ pt.Mv(*pt).Mv ◦

```
package main

import "fmt"

type Foo struct {
    Bar int
}

func (f Foo) Increment() {
    f.Bar += 1
}

func main() {
    var p *Foo

    // Calling `p.Increment` is automatically changed to `(*p).Increment` by the compiler.
```

```
    // (Note that `*p` is going to remain at 0 because a copy of `*p`, and not the original
`*p` are being edited)
    p = &Foo{}
    fmt.Printf("(*p).Bar is %d\n", (*p).Bar)
    p.Increment()
    fmt.Printf("(*p).Bar is %d\n", (*p).Bar)

    // As you can see, calling `(*p).Increment` directly does the same thing.
    p = &Foo{}
    fmt.Printf("(*p).Bar is %d\n", (*p).Bar)
    (*p).Increment()
    fmt.Printf("(*p).Bar is %d\n", (*p).Bar)
}
```

Go Spec Pointers v.ValueEffective Go 。

---

*1.Bar*p&f ()* 。

2 。

*。

```
package main

import (
    "fmt"
)

type Person struct {
    Name string
}

func main() {
    c := new(Person) // returns pointer
    c.Name = "Catherine"
    fmt.Println(c.Name) // prints: Catherine
    d := c
    d.Name = "Daniel"
    fmt.Println(c.Name) // prints: Daniel
    // Adding an Asterix before a pointer dereferences the pointer
    i := *d
    i.Name = "Ines"
    fmt.Println(c.Name) // prints: Daniel
    fmt.Println(d.Name) // prints: Daniel
    fmt.Println(i.Name) // prints: Ines
}
```

。 。 func copy(dst, src []Type) int 。

```
package main

import (
    "fmt"
)
```

```
func main() {
    x := []byte{'a', 'b', 'c'}
    fmt.Printf("%s", x)       // prints: abc
    y := x
    y[0], y[1], y[2] = 'x', 'y', 'z'
    fmt.Printf("%s", x)       // prints: xyz
    z := make([]byte, len(x))
    // To copy the value to another slice, but
    // but not the memory address use copy:
    _ = copy(z, x)            // returns count of items copied
    fmt.Printf("%s", z)       // prints: xyz
    z[0], z[1], z[2] = 'a', 'b', 'c'
    fmt.Printf("%s", x)       // prints: xyz
    fmt.Printf("%s", z)       // prints: abc
}
```

```
func swap(x, y *int) {
  *x, *y = *y, *x
}

func main() {
  x := int(1)
  y := int(2)
  // variable addresses
  swap(&x, &y)
  fmt.Println(x, y)
}
```

https://riptutorial.com/zh-CN/go/topic/1239/

49:

Go。 。 。

Examples

Go。 。

```
type Painter interface {
    Paint()
}
```

。 。

```
type Rembrandt struct{}

func (r Rembrandt) Paint() {
    // use a lot of canvas here
}
```

。

```
var p Painter
p = Rembrandt{}
```

。 。

```
type Singer interface {
     Sing()
}

type Writer interface {
     Write()
}

type Human struct{}

func (h *Human) Sing() {
    fmt.Println("singing")
}

func (h *Human) Write() {
    fmt.Println("writing")
}



type OnlySinger struct{}
func (o *OnlySinger) Sing() {
    fmt.Println("singing")
}
```

HumanSingerWriterOnlySingerSinger◦

◦ interface{} ◦ type◦ ◦

```
var a interface{}
var i int = 5
s := "Hello world"

type StructType struct {
    i, j int
    k string
}


// all are valid statements
a = i
a = s
a = &StructType{1, 2, "hello"}
```

◦ ◦

```
i = a.(int)
s = a.(string)
m := a.(*StructType)
```

```
i, ok := a.(int)
s, ok := a.(string)
m, ok := a.(*StructType)
```

okinterface a◦ okfalse ◦

---

interface Singer h HumanOnlySinger.OnlySinger.Singer◦

```
var h Singer
h = &human{}

h.Sing()
```

go◦ ◦

```
type Rembrandt struct{}

func (r Rembrandt) Paint() {}

type Picasso struct{}

func (r Picasso) Paint() {}
```

Painter

```
type Painter interface {
```

```
    Paint()
}
```

```
func WhichPainter(painter Painter) {
    switch painter.(type) {
    case Rembrandt:
        fmt.Println("The underlying type is Rembrandt")
    case Picasso:
        fmt.Println("The underlying type is Picasso")
    default:
        fmt.Println("Unknown type")
    }
}
```

“”。 。

TITT。

```
type T struct{}

var _ I = T{}       // Verify that T implements I.
var _ I = (*T)(nil) // Verify that *T implements I.
```

T*TI 。

FAQ

```
func convint(v interface{}) (int,error) {
    switch u := v.(type) {
    case int:
        return u, nil
    case float64:
        return int(u), nil
    case string:
        return strconv(u)
    default:
        return 0, errors.New("Unsupported type")
    }
}
```

Type Assertion。

```
interfaceVariable.(DataType)
```

Subberstruct MyType Subber

```
package main

import (
    "fmt"
)

type Subber interface {
```

```
    Sub(a, b int) int
}

type MyType struct {
    Msg string
}

//Implement method Sub(a,b int) int
func (m *MyType) Sub(a, b int) int {
    m.Msg = "SUB!!!"

    return a - b;
}

func main() {
    var interfaceVar Subber = &MyType{}
    fmt.Println(interfaceVar.Sub(6,5))
    fmt.Println(interfaceVar.(*MyType).Msg)
}
```

`.(*MyType)Msg` Field。 `interfaceVar.Msg`

```
interfaceVar.Msg undefined (type Subber has no field or method Msg)
```

。 ABC.....{abcde}。 xZ“xZ”。 。 。 {0,1,2,3,4,5...}。 {a $_n$ | a $_0$ = 0a $_n$ = a $_{n-1}$ +1}。 $_0$ = 0 $_n$ = a $_{n-1}$ + 1 。 。 。 {0,2,4,6,8,10...}。 。 {n | n n2}。 。 3137871。

Go。

```
type Number interface {
    IsNumber() bool // the implementation filter "meysam" from 3.14, 2 and 3
}

type NaturalNumber interface {
    Number
    IsNaturalNumber() bool // the implementation filter 3.14 from 2 and 3
}

type EvenNumber interface {
    NaturalNumber
    IsEvenNumber() bool // the implementation filter 3 from 2
}
```

`NumberIsNumberNaturalNumber IsNumberIsNaturalNumberEvenNumber IsNumber IsNaturalNumber IsEvenNumber。 interface{}。`

https://riptutorial.com/zh-CN/go/topic/1221/

50: I / O.

Examples

scanf

Scanf。。。。c。

```
# Read integer
var i int
fmt.Scanf("%d", &i)

# Read string
var str string
fmt.Scanf("%s", &str)
```

scan

。。。。

```
# Read integer
var i int
fmt.Scan(&i)

# Read string
var str string
fmt.Scan(&str)
```

scanln

SscanlnSscanEOF。

```
# Read string
var input string
fmt.Scanln(&input)
```

bufio

```
# Read using Reader
reader := bufio.NewReader(os.Stdin)
text, err := reader.ReadString('\n')

# Read using Scanner
scanner := bufio.NewScanner(os.Stdin)
for scanner.Scan() {
    fmt.Println(scanner.Text())
}
```

I / O. https://riptutorial.com/zh-CN/go/topic/9741/i---o-

51:

defer◦ ◦ ◦

- someFuncargs
- defer func{// code goes here}

Defer defer◦ SIGKILL ◦

Examples

Go◦ Deferdefer◦

```
defer someFunction()
```

defer◦

- return
-
-

```
func main() {
    fmt.Println("First main statement")
    defer logExit("main") // position of defer statement here does not matter
    fmt.Println("Last main statement")
}

func logExit(name string) {
    fmt.Printf("Function %s returned\n", name)
}
```

```
First main statement
Last main statement
Function main returned
```

◦ deferdefer defer S

```
func main() {
    defer logNum(1)
    fmt.Println("First main statement")
    defer logNum(2)
    defer logNum(3)
    panic("panic occurred")
    fmt.Println("Last main statement") // not printed
    defer logNum(3) // not deferred since execution flow never reaches this line
}

func logNum(i int) {
    fmt.Printf("Num %d\n", i)
}
```

```
First main statement
Num 3
Num 2
Num 1
panic: panic occurred

goroutine 1 [running]:
....
```

defer

```
func main() {
    i := 1
    defer logNum(i) // deferred function call: logNum(1)
    fmt.Println("First main statement")
    i++
    defer logNum(i) // deferred function call: logNum(2)
    defer logNum(i*i) // deferred function call: logNum(4)
    return // explicit return
}

func logNum(i int) {
    fmt.Printf("Num %d\n", i)
}
```

```
First main statement
Num 4
Num 2
Num 1
```

```
func main() {
    fmt.Println(plusOne(1)) // 2
    return
}

func plusOne(i int) (result int) { // overkill! only for demonstration
    defer func() {result += 1}() // anonymous function must be called by adding ()

    // i is returned as result, which is updated by deferred function above
    // after execution of below return
    return i
}
```

defer◦

-
-
-
- waitgroup `defer wg.Done()`

◦

```
resp, err := http.Get(url)
if err != nil {
return err
}
```

```
defer resp.Body.Close() // Body will always get closed
```

Javafinally◦ ◦ defergoroutinego◦ gogo◦

```
func MyFunc() {
    conn := GetConnection()    // Some kind of connection that must be closed.
    defer conn.Close()        // Will be executed when MyFunc returns, regardless of how.
    // Do some things...
    if someCondition {
        return                // conn.Close() will be called
    }
    // Do more things
}// Implicit return - conn.Close() will still be called
```

conn.Close()conn.Close - ◦ ◦ - parens

```
defer func(){
    // Do some cleanup
}()
```

https://riptutorial.com/zh-CN/go/topic/2795/

52:

GoGo。

Go 1.8Linux。

Examples

```
package main

import "fmt"

var V int

func F() { fmt.Printf("Hello, number %d\n", V) }
```

```
go build -buildmode=plugin
```

```
p, err := plugin.Open("plugin_name.so")
if err != nil {
    panic(err)
}

v, err := p.Lookup("V")
if err != nil {
    panic(err)
}

f, err := p.Lookup("F")
if err != nil {
    panic(err)
}

*v.(*int) = 7
f.(func())() // prints "Hello, number 7"
```

2017。

https://riptutorial.com/zh-CN/go/topic/9150/

53:

- func Notifyc chan < - os.Signalsig ... os.Signal

c chan < - os.Signal	`os.Signalchannel ;sigChan := make(chan os.Signal)sigChan := make(chan os.Signal)`
sig ... os.Signal	`os.Signalchannel` ◦ https://golang.org/pkg/syscall/#pkg-constants ◦

Examples

◦ os/signal◦ ◦

```
package main

import (
    "fmt"
    "os"
    "os/signal"
)

func main() {
    // create a channel for os.Signal
    sigChan := make(chan os.Signal)

    // assign all signal notifications to the channel
    signal.Notify(sigChan)

    // blocks until you get a signal from the OS
    select {
    // when a signal is received
    case sig := <-sigChan:
        // print this line telling us which signal was seen
        fmt.Println("Received signal from OS:", sig)
    }
}
```

◦

```
$ go run signals.go
^CReceived signal from OS: interrupt
```

^CCTRL+C SIGINT◦

https://riptutorial.com/zh-CN/go/topic/4497/

54:

。

GoArrays“”

- var variableName [5] ArrayType //5。
- var variableName [2] [3] ArrayType = {{Value1Value2Value3}{Value4Value5Value6}} //
- variableName= [...] ArrayType {Value1Value2Value3} //3
- arrayName [2] //。
- arrayName [5] = 0 //index。
- arrayName [0] //
- arrayName [lenarrayName-1] //

Examples

go。

`[] 。`

`var array = [size]Type size 42`42`TypeType intstring`

Go。

```
// Creating arrays of 6 elements of type int,
// and put elements 1, 2, 3, 4, 5 and 6 inside it, in this exact order:
var array1 [6]int = [6]int {1, 2, 3, 4, 5, 6} // classical way
var array2 = [6]int {1, 2, 3, 4, 5, 6} // a less verbose way
var array3 = [...]int {1, 2, 3, 4, 5, 6} // the compiler will count the array elements by
itself

fmt.Println("array1:", array1) // > [1 2 3 4 5 6]
fmt.Println("array2:", array2) // > [1 2 3 4 5 6]
fmt.Println("array3:", array3) // > [1 2 3 4 5 6]


// Creating arrays with default values inside:
zeros := [8]int{}               // Create a list of 8 int filled with 0
ptrs := [8]*int{}               // a list of int pointers, filled with 8 nil references (
<nil> )
emptystr := [8]string{}         // a list of string filled with 8 times ""

fmt.Println("zeroes:", zeros)      // > [0 0 0 0 0 0 0 0]
fmt.Println("ptrs:", ptrs)         // > [<nil> <nil> <nil> <nil> <nil> <nil> <nil> <nil>]
fmt.Println("emptystr:", emptystr) // > [       ]
// values are empty strings, separated by spaces,
// so we can just see separating spaces


// Arrays are also working with a personalized type
type Data struct {
    Number int
    Text   string
```

```
}

// Creating an array with 8 'Data' elements
// All the 8 elements will be like {0, ""} (Number = 0, Text = "")
structs := [8]Data{}

fmt.Println("structs:", structs) // > [{0 } {0 } {0 } {0 } {0 } {0 } {0 } {0 }]
// prints {0 } because Number are 0 and Text are empty; separated by a space
```

□□□□□

。

`[sizeDim1][sizeDim2]..[sizeLastDim]type` sizeDimtype。

`[2][3]int`**2 3int** 。
2 3。

`var values := [2017][12][31][24][60]intvar values := [2017][12][31][24][60]int`0。

19:422016-01-31`values[2016][0][30][19][42]` **0**1

```
// Defining a 2d Array to represent a matrix like
// 1 2 3     So with 2 lines and 3 columns;
// 4 5 6
var multiDimArray := [2/*lines*/][3/*columns*/]int{ [3]int{1, 2, 3}, [3]int{4, 5, 6} }

// That can be simplified like this:
var simplified := [2][3]int{{1, 2, 3}, {4, 5, 6}}

// What does it looks like ?
fmt.Println(multiDimArray)
// > [[1 2 3] [4 5 6]]

fmt.Println(multiDimArray[0])
// > [1 2 3]    (first line of the array)

fmt.Println(multiDimArray[0][1])
// > 2          (cell of line 0 (the first one), column 1 (the 2nd one))
```

```
// We can also define array with as much dimensions as we need
// here, initialized with all zeros
var multiDimArray := [2][4][3][2]string{}

fmt.Println(multiDimArray);
// Yeah, many dimensions stores many data
// > [[[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]]
//   [[[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]]
```

```
// We can set some values in the array's cells
multiDimArray[0][0][0][0] := "All zero indexes"  // Setting the first value
```

```
multiDimArray[1][3][2][1] := "All indexes to max"  // Setting the value at extreme location

fmt.Println(multiDimArray);
// If we could see in 4 dimensions, maybe we could see the result as a simple format

// > [[[["All zero indexes" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]]
//   [[[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" ""]]]
//    [[["" ""] ["" ""]] [["" ""] ["" ""]] [["" ""] ["" "All indexes to max"]]]]
```

。 。

0-1。

`arrayName[index]` “index”。

```
var array = [6]int {1, 2, 3, 4, 5, 6}

fmt.Println(array[-42]) // invalid array index -1 (index must be non-negative)
fmt.Println(array[-1]) // invalid array index -1 (index must be non-negative)
fmt.Println(array[0]) // > 1
fmt.Println(array[1]) // > 2
fmt.Println(array[2]) // > 3
fmt.Println(array[3]) // > 4
fmt.Println(array[4]) // > 5
fmt.Println(array[5]) // > 6
fmt.Println(array[6]) // invalid array index 6 (out of bounds for 6-element array)
fmt.Println(array[42]) // invalid array index 42 (out of bounds for 6-element array)
```

。

```
var array = [6]int {1, 2, 3, 4, 5, 6}

fmt.Println(array) // > [1 2 3 4 5 6]

array[0] := 6
fmt.Println(array) // > [6 2 3 4 5 6]

array[1] := 5
fmt.Println(array) // > [6 5 3 4 5 6]

array[2] := 4
fmt.Println(array) // > [6 5 4 4 5 6]

array[3] := 3
fmt.Println(array) // > [6 5 4 3 5 6]

array[4] := 2
fmt.Println(array) // > [6 5 4 3 2 6]

array[5] := 1
fmt.Println(array) // > [6 5 4 3 2 1]
```

https://riptutorial.com/zh-CN/go/topic/390/

55: I / O.

- fileerr= os.Open *name* //。。
- fileerr= os.Create *name* //。。。
- fileerr= os.OpenFile *name* *flags* *perm* //。。
- dataerr= ioutil.ReadFile *name* //。。
- err= ioutil.WriteFile *name* *data* *perm* //UNIX。。
- err= os.Remove *name* //。。
- err= os.RemoveAll *name* //。。
- ERR= os.Rename *oldNamenewName* //。。

。 "hello.txt"。
error。 nil。
os*os.File。 io.ReadWriter Read(data)Write(data)。。
[]byte。
UNIXos.FileMode。。
int。。 os.O_RDONLY os.O_WRONLY os.O_RDWR os.O_APPEND os.O_CREATE os.O_EXCL os.O_SYNC os.O_TRUNC。

Examples

ioutil

“Helloworld” to `test.txt`。 I / O。

```
package main

import (
    "fmt"
    "io/ioutil"
)

func main() {
    hello := []byte("Hello, world!")

    // Write `Hello, world!` to test.txt that can read/written by user and read by others
    err := ioutil.WriteFile("test.txt", hello, 0644)
    if err != nil {
        panic(err)
    }

    // Read test.txt
```

```
    data, err := ioutil.ReadFile("test.txt")
    if err != nil {
        panic(err)
    }

    // Should output: `The file contains: Hello, world!`
    fmt.Println("The file contains: " + string(data))
}
```

```
package main

import (
    "fmt"
    "io/ioutil"
)

func main() {
    files, err := ioutil.ReadDir(".")
    if err != nil {
        panic(err)
    }

    fmt.Println("Files and folders in the current directory:")

    for _, fileInfo := range files {
        fmt.Println(fileInfo.Name())
    }
}
```

```
package main

import (
    "fmt"
    "io/ioutil"
)

func main() {
    files, err := ioutil.ReadDir(".")
    if err != nil {
        panic(err)
    }

    fmt.Println("Folders in the current directory:")

    for _, fileInfo := range files {
        if fileInfo.IsDir() {
            fmt.Println(fileInfo.Name())
        }
    }
}
```

I / O. https://riptutorial.com/zh-CN/go/topic/1033/i---o-

56: + HTML

Examples

{{.}}◦

```
package main

import (
    "fmt"
    "os"
    "text/template"
)

func main() {
    const (
        letter = `Dear {{.}}, How are you?`
    )

    tmpl, err := template.New("letter").Parse(letter)
    if err != nil {
        fmt.Println(err.Error())
    }

    tmpl.Execute(os.Stdout, "Professor Jones")
}
```

```
Dear Professor Jones, How are you?
```

{{range .}}{{end}}◦

```
package main

import (
    "fmt"
    "os"
    "text/template"
)

func main() {
    const (
        letter = `Dear {{range .}}{{.}}, {{end}} How are you?`
    )

    tmpl, err := template.New("letter").Parse(letter)
    if err != nil {
        fmt.Println(err.Error())
    }

    tmpl.Execute(os.Stdout, []string{"Harry", "Jane", "Lisa", "George"})
}
```

```
Dear Harry, Jane, Lisa, George,  How are you?
```

funcMapFuncs()。 increment()。 。

A -{{-end -}}。

```
package main

import (
    "fmt"
    "os"
    "text/template"
)

var funcMap = template.FuncMap{
    "increment": increment,
}

func increment(x int) int {
    return x + 1
}

func main() {
    const (
        letter = `Dear {{with $names := .}}
        {{- range $i, $val := $names}}
            {{- if lt (increment $i) (len $names)}}
                {{- $val}}, {{else -}} and {{$val}}{{end}}
        {{- end}}{{end}}; How are you?`
    )

    tmpl, err := template.New("letter").Funcs(funcMap).Parse(letter)
    if err != nil {
        fmt.Println(err.Error())
    }

    tmpl.Execute(os.Stdout, []string{"Harry", "Jane", "Lisa", "George"})
}
```

```
Dear Harry, Jane, Lisa, and George; How are you?
```

{{.FieldName}}。

```
package main

import (
    "fmt"
    "os"
    "text/template"
)

type Person struct {
    FirstName string
    LastName  string
    Street    string
    City      string
    State     string
    Zip       string
}
```

```
func main() {
    const (
        letter = `------------------------------
{{range .}}{{.FirstName}} {{.LastName}}
{{.Street}}
{{.City}}, {{.State}} {{.Zip}}

Dear {{.FirstName}},
    How are you?

------------------------------
{{end}}`
    )

    tmpl, err := template.New("letter").Parse(letter)
    if err != nil {
        fmt.Println(err.Error())
    }

    harry := Person{
        FirstName: "Harry",
        LastName:  "Jones",
        Street:    "1234 Main St.",
        City:      "Springfield",
        State:     "IL",
        Zip:       "12345-6789",
    }

    jane := Person{
        FirstName: "Jane",
        LastName:  "Sherman",
        Street:    "8511 1st Ave.",
        City:      "Dayton",
        State:     "OH",
        Zip:       "18515-6261",
    }

    tmpl.Execute(os.Stdout, []Person{harry, jane})
}
```

```
------------------------------
Harry Jones
1234 Main St.
Springfield, IL 12345-6789

Dear Harry,
    How are you?

------------------------------
Jane Sherman
8511 1st Ave.
Dayton, OH 18515-6261

Dear Jane,
    How are you?

------------------------------
```

HTML

。

```
package main

import (
    "fmt"
    "html/template"
    "os"
)

type Person struct {
    FirstName string
    LastName  string
    Street    string
    City      string
    State     string
    Zip       string
    AvatarUrl string
}

func main() {
    const (
        letter = `<html><body><table>
<tr><th></th><th>Name</th><th>Address</th></tr>
{{range .}}
<tr>
<td><img src="{{.AvatarUrl}}"></td>
<td>{{.FirstName}} {{.LastName}}</td>
<td>{{.Street}}, {{.City}}, {{.State}} {{.Zip}}</td>
</tr>
{{end}}
</table></body></html>`
    )

    tmpl, err := template.New("letter").Parse(letter)
    if err != nil {
        fmt.Println(err.Error())
    }

    harry := Person{
        FirstName: "Harry",
        LastName:  "Jones",
        Street:    "1234 Main St.",
        City:      "Springfield",
        State:     "IL",
        Zip:       "12345-6789",
        AvatarUrl: "harry.png",
    }

    jane := Person{
        FirstName: "Jane",
        LastName:  "Sherman",
        Street:    "8511 1st Ave.",
        City:      "Dayton",
        State:     "OH",
        Zip:       "18515-6261",
        AvatarUrl: "jane.png",
    }
```

```
    tmpl.Execute(os.Stdout, []Person{harry, jane})
}
```

```
<html><body><table>
<tr><th></th><th>Name</th><th>Address</th></tr>

<tr>
<td><img src="harry.png"></td>
<td>Harry Jones</td>
<td>1234 Main St., Springfield, IL 12345-6789</td>
</tr>

<tr>
<td><img src="jane.png"></td>
<td>Jane Sherman</td>
<td>8511 1st Ave., Dayton, OH 18515-6261</td>
</tr>

</table></body></html>
```

HTML

`text/template`HTML。 Harry`FirstName`。

```
package main

import (
    "fmt"
    "html/template"
    "os"
)

type Person struct {
    FirstName string
    LastName  string
    Street    string
    City      string
    State     string
    Zip       string
    AvatarUrl string
}

func main() {
    const (
        letter = `<html><body><table>
<tr><th></th><th>Name</th><th>Address</th></tr>
{{range .}}
<tr>
<td><img src="{{.AvatarUrl}}"></td>
<td>{{.FirstName}} {{.LastName}}</td>
<td>{{.Street}}, {{.City}}, {{.State}} {{.Zip}}</td>
</tr>
{{end}}
</table></body></html>`
    )

    tmpl, err := template.New("letter").Parse(letter)
```

```
    if err != nil {
        fmt.Println(err.Error())
    }

    harry := Person{
        FirstName: `Harry<script>alert("You've been hacked!")</script>`,
        LastName:  "Jones",
        Street:    "1234 Main St.",
        City:      "Springfield",
        State:     "IL",
        Zip:       "12345-6789",
        AvatarUrl: "harry.png",
    }

    jane := Person{
        FirstName: "Jane",
        LastName:  "Sherman",
        Street:    "8511 1st Ave.",
        City:      "Dayton",
        State:     "OH",
        Zip:       "18515-6261",
        AvatarUrl: "jane.png",
    }

    tmpl.Execute(os.Stdout, []Person{harry, jane})
}
```

```
<html><body><table>
<tr><th></th><th>Name</th><th>Address</th></tr>

<tr>
<td><img src="harry.png"></td>
<td>Harry<script>alert("You've been hacked!")</script> Jones</td>
<td>1234 Main St., Springfield, IL 12345-6789</td>
</tr>

<tr>
<td><img src="jane.png"></td>
<td>Jane Sherman</td>
<td>8511 1st Ave., Dayton, OH 18515-6261</td>
</tr>

</table></body></html>
```

◦ html/templatetext/template

```
<html><body><table>
<tr><th></th><th>Name</th><th>Address</th></tr>

<tr>
<td><img src="harry.png"></td>
<td>Harry&amp;lt;script&amp;gt;alert(&amp;#34;You&amp;#39;ve been hacked!&amp;#34;)&amp;lt;/script&amp;gt; Jones</td>
<td>1234 Main St., Springfield, IL 12345-6789</td>
</tr>

<tr>
<td><img src="jane.png"></td>
<td>Jane Sherman</td>
<td>8511 1st Ave., Dayton, OH 18515-6261</td>
```

```
</tr>

</table></body></html>
```

。

+ HTML https://riptutorial.com/zh-CN/go/topic/3888/plus-html

57:

- funct TexampleOnei intn int{return i} //struct
- funct * TexampleTwoi intn int{return i} //struct

Examples

Go。

。

```
package main

import (
    "fmt"
)

type Employee struct {
    Name string
    Age  int
    Rank int
}

func (empl *Employee) Promote() {
    empl.Rank++
}

func main() {

    Bob := new(Employee)

    Bob.Rank = 1
    fmt.Println("Bobs rank now is: ", Bob.Rank)
    fmt.Println("Lets promote Bob!")

    Bob.Promote()

    fmt.Println("Now Bobs rank is: ", Bob.Rank)

}
```

```
Bobs rank now is:  1
Lets promote Bob!
Now Bobs rank is:  2
```

golang“”

```
package main

import (
    "fmt"
)
```

```
type Employee struct {
    Name string
    Age  int
    Rank int
}

func (empl *Employee) Promote() *Employee {
    fmt.Printf("Promoting %s\n", empl.Name)
    empl.Rank++
    return empl
}

func (empl *Employee) SetName(name string) *Employee {
    fmt.Printf("Set name of new Employee to %s\n", name)
    empl.Name = name
    return empl
}

func main() {

    worker := new(Employee)

    worker.Rank = 1

    worker.SetName("Bob").Promote()

    fmt.Printf("Here we have %s with rank %d\n", worker.Name, worker.Rank)

}
```

```
Set name of new Employee to Bob
Promoting Bob
Here we have Bob with rank 2
```

Increment-Decrement

Go++ - c / c ++。

```
    package main

    import (
        "fmt"
    )

    func abcd(a int, b int) {
     fmt.Println(a," ",b)
    }
    func main() {
        a:=5
        abcd(a++,++a)
    }
```

++

https://riptutorial.com/zh-CN/go/topic/3890/

58:

Go `time`。

`time.Time` 。

- time.Date2016time.December31,23,59,59,999time.UTC//
- date1 == date2 //2`true`
- date1= date2 //2`true`
- date1.Beforedate2//`true`
- date1.Afterdate2//`true`

Examples

。

```
const timeFormat = "15 Monday January 2006"

func ParseDate(s string) (time.Time, error) {
    t, err := time.Parse(timeFormat, s)
    if err != nil {
        // time.Time{} returns January 1, year 1, 00:00:00.000000000 UTC
        // which according to the source code is the zero value for time.Time
        // https://golang.org/src/time/time.go#L23
        return time.Time{}, err
    }
    return t, nil
}
```

。 `time.Parse` 。

```
//           time.Parse(   format   , date to parse)
date, err := time.Parse("01/02/2006",  "04/08/2017")
if err != nil {
    panic(err)
}

fmt.Println(date)
// Prints 2017-04-08 00:00:00 +0000 UTC
```

。 `01/02/2006MM/DD/YYYY` 。

`Mon Jan 2 15:04:05 -0700 MST 2006`;。 。

。

```
const (
    stdLongMonth             // "January"
    stdMonth                 // "Jan"
    stdNumMonth              // "1"
```

```
    stdZeroMonth             // "01"
    stdLongWeekDay           // "Monday"
    stdWeekDay               // "Mon"
    stdDay                   // "2"
    stdUnderDay              // "_2"
    stdZeroDay               // "02"
    stdHour                  // "15"
    stdHour12                // "3"
    stdZeroHour12            // "03"
    stdMinute                // "4"
    stdZeroMinute            // "04"
    stdSecond                // "5"
    stdZeroSecond            // "05"
    stdLongYear              // "2006"
    stdYear                  // "06"
    stdPM                    // "PM"
    stdpm                    // "pm"
    stdTZ                    // "MST"
    stdISO8601TZ             // "Z0700"  // prints Z for UTC
    stdISO8601SecondsTZ      // "Z070000"
    stdISO8601ShortTZ        // "Z07"
    stdISO8601ColonTZ        // "Z07:00" // prints Z for UTC
    stdISO8601ColonSecondsTZ // "Z07:00:00"
    stdNumTZ                 // "-0700"  // always numeric
    stdNumSecondsTz          // "-070000"
    stdNumShortTZ            // "-07"    // always numeric
    stdNumColonTZ            // "-07:00" // always numeric
    stdNumColonSecondsTZ     // "-07:00:00"
)
```

2。

Go 4

- `date1 == date2` 2`true`
- `date1 != date2` 2`true`
- `date1.Before(date2)` `true`
- `date1.After(date2)` `true`

  2 Time to compare `AfterBeforefalse`

  - `date1 == date1` `true`
  - `date1 != date1` `false`
  - `date1.After(date1)` `false`
  - `date1.Before(date1)` `false`

  4

  - `date1 == date2 && date1.After(date2)` date1date2`true`
    `! (date1.Before(date2))`
  - `date1 == date2 && date1.Before(date2)` date1date2date2`!(date1.After(date2))true`

```
// Init 2 dates for example
var date1 = time.Date(2009, time.November, 10, 23, 0, 0, 0, time.UTC)
var date2 = time.Date(2017, time.July, 25, 16, 22, 42, 123, time.UTC)
```

```
var date3 = time.Date(2017, time.July, 25, 16, 22, 42, 123, time.UTC)

bool1 := date1.Before(date2) // true, because date1 is before date2
bool2 := date1.After(date2) // false, because date1 is not after date2

bool3 := date2.Before(date1) // false, because date2 is not before date1
bool4 := date2.After(date1) // true, because date2 is after date1

bool5 := date1 == date2 // false, not the same moment
bool6 := date1 == date3 // true, different objects but representing the exact same time

bool7 := date1 != date2 // true, different moments
bool8 := date1 != date3 // false, not different moments

bool9 := date1.After(date3) // false, because date1 is not after date3 (that are the same)
bool10:= date1.Before(date3) // false, because date1 is not before date3 (that are the same)

bool11 := !(date1.Before(date3)) // true, because date1 is not before date3
bool12 := !(date1.After(date3)) // true, because date1 is not after date3
```

https://riptutorial.com/zh-CN/go/topic/8860/

59:

- t= template.Parse `{{.MyName .MyAge}}`
- t.Executeos.Stdoutstruct {MyValueMyAge string} {“John Doe”“40.1”}

Golang

1. `text/template`
2. `html/template`

HTML。

Examples

struct

```
package main

import (
    "log"
    "text/template"
    "os"
)

type Person struct{
    MyName string
    MyAge int
}

var myTempContents string= `
This person's name is : {{.MyName}}
And he is {{.MyAge}} years old.
`

func main() {
    t,err := template.New("myTemp").Parse(myTempContents)
    if err != nil{
        log.Fatal(err)
    }
    myPersonSlice := []Person{ {"John Doe",41},{"Peter Parker",17} }
    for _,myPerson := range myPersonSlice{
        t.Execute(os.Stdout,myPerson)
    }
}
```

```
package main

import (
    "fmt"
    "net/http"
    "os"
```

```
    "text/template"
)

var requestTemplate string = `
{{range $i, $url := .URLs}}
{{ $url }} {{(status_code $url)}}
{{ end }}`

type Requests struct {
    URLs []string
}

func main() {
    var fns = template.FuncMap{
        "status_code": func(x string) int {
            resp, err := http.Head(x)
            if err != nil {
                return -1
            }
            return resp.StatusCode
        },
    }

    req := new(Requests)
    req.URLs = []string{"http://godoc.org", "http://stackoverflow.com", "http://linux.org"}

    tmpl := template.Must(template.New("getBatch").Funcs(fns).Parse(requestTemplate))
    err := tmpl.Execute(os.Stdout, req)
    if err != nil {
        fmt.Println(err)
    }
}
```

status_code◦

```
http://godoc.org 200

http://stackoverflow.com 200

http://linux.org 200
```

https://riptutorial.com/zh-CN/go/topic/1402/

60:

mgomangoGoMongoDBGoAPI。

API

[https://gopkg.in/mgo.v2][1]

Examples

```
package main

import (
        "fmt"
        "log"
        "gopkg.in/mgo.v2"
        "gopkg.in/mgo.v2/bson"
)

type Person struct {
        Name string
        Phone string
}

func main() {
        session, err := mgo.Dial("server1.example.com,server2.example.com")
        if err != nil {
                panic(err)
        }
        defer session.Close()

        // Optional. Switch the session to a monotonic behavior.
        session.SetMode(mgo.Monotonic, true)

        c := session.DB("test").C("people")
        err = c.Insert(&Person{"Ale", "+55 53 8116 9639"},
                   &Person{"Cla", "+55 53 8402 8510"})
        if err != nil {
                log.Fatal(err)
        }

        result := Person{}
        err = c.Find(bson.M{"name": "Ale"}).One(&result)
        if err != nil {
                log.Fatal(err)
        }

        fmt.Println("Phone:", result.Phone)
}
```

https://riptutorial.com/zh-CN/go/topic/8898/

61:

Go。。

Examples

main.go

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println(Sum(4,5))
}

func Sum(a, b int) int {
    return a + b
}
```

main_test.go

```
package main

import (
    "testing"
)

// Test methods start with `Test`
func TestSum(t *testing.T) {
    got := Sum(1, 2)
    want := 3
    if got != want {
        t.Errorf("Sum(1, 2) == %d, want %d", got, want)
    }
}
```

go test

```
$ go test
ok      test_app    0.005s
```

-v

```
$ go test -v
=== RUN   TestSum
--- PASS: TestSum (0.00s)
PASS
ok      _/tmp    0.000s
```

./ ./...

```
$ go test -v ./...
ok      github.com/me/project/dir1    0.008s
=== RUN   TestSum
--- PASS: TestSum (0.00s)
PASS
ok      github.com/me/project/dir2    0.008s
=== RUN   TestDiff
--- PASS: TestDiff (0.00s)
PASS
```

```
go test -v -run=<TestName> // will execute only test with this name
```

```
go test -v run=TestSum
```

sum.go

```
package sum

// Sum calculates the sum of two integers
func Sum(a, b int) int {
    return a + b
}
```

sum_test.go

```
package sum

import "testing"

func BenchmarkSum(b *testing.B) {
    for i := 0; i < b.N; i++ {
        _ = Sum(2, 3)
    }
}
```

```
$ go test -bench=.
BenchmarkSum-8    2000000000             0.49 ns/op
ok      so/sum    1.027s
```

。

main.go

```
package main

import (
    "fmt"
)
```

```
func main() {
    fmt.Println(Sum(4, 5))
}

func Sum(a, b int) int {
    return a + b
}
```

9 ◦ Sum◦ main_test.gomain.go

```
package main

import (
    "testing"
)

// Test methods start with Test
func TestSum(t *testing.T) {
    // Note that the data variable is of type array of anonymous struct,
    // which is very handy for writing table-driven unit tests.
    data := []struct {
        a, b, res int
    }{
        {1, 2, 3},
        {0, 0, 0},
        {1, -1, 0},
        {2, 3, 5},
        {1000, 234, 1234},
    }

    for _, d := range data {
        if got := Sum(d.a, d.b); got != d.res {
            t.Errorf("Sum(%d, %d) == %d, want %d", d.a, d.b, got, d.res)
        }
    }
}
```

◦ ◦

◦ ◦

sum.go

```
package sum

// Sum calculates the sum of two integers
func Sum(a, b int) int {
    return a + b
}
```

sum_test.go

```
package sum

import "fmt"
```

```
func ExampleSum() {
    x := Sum(1, 2)
    fmt.Println(x)
    fmt.Println(Sum(-1, -1))
    fmt.Println(Sum(0, 0))

    // Output:
    // 3
    // -2
    // 0
}
```

go testgo test sumgo test ./sum ◦

```
ok      so/sum    0.005s
```

```
func ExampleSum_fail() {
    x := Sum(1, 2)
    fmt.Println(x)

    // Output:
    // 5
}
```

go test

```
$ go test
--- FAIL: ExampleSum_fail (0.00s)
got:
3
want:
5
FAIL
exit status 1
FAIL    so/sum    0.006s
```

sum -

```
go doc -http=:6060
```

http// localhost6060 / pkg / FOLDER / sum / *FOLDER*sumso ◦ sum

Package sum

```
import "so/sum"
```

Overview
Index
Examples

Overview ▾

Package sum is a sample package for test purpos

Index ▾

func Sum(a, b int) int

Examples

Sum

Package files

sum.go

- tearDown。

。

```
// Standard numbers map
var numbers map[string]int = map[string]int{"zero": 0, "three": 3}

// TestMain will exec each test, one by one
func TestMain(m *testing.M) {
    // exec setUp function
    setUp("one", 1)
    // exec test and this returns an exit code to pass to os
    retCode := m.Run()
    // exec tearDown function
    tearDown("one")
    // If exit code is distinct of zero,
    // the test will be failed (red)
    os.Exit(retCode)
}

// setUp function, add a number to numbers slice
func setUp(key string, value int) {
    numbers[key] = value
}

// tearDown function, delete a number to numbers slice
func tearDown(key string) {
    delete(numbers, key)
}

// First test
func TestOnePlusOne(t *testing.T) {
    numbers["one"] = numbers["one"] + 1

    if numbers["one"] != 2 {
        t.Error("1 plus 1 = 2, not %v", value)
    }
}

// Second test
func TestOnePlusTwo(t *testing.T) {
    numbers["one"] = numbers["one"] + 2

    if numbers["one"] != 3 {
        t.Error("1 plus 2 = 3, not %v", value)
    }
}
```

```
 // ID of Person will be saved in database
personID := 12345
// Name of Person will be saved in database
personName := "Toni"

func TestMain(m *testing.M) {
    // You create an Person and you save in database
    setUp(&Person{
            ID:   personID,
            Name: personName,
            Age:  19,
```

```
        })
    retCode := m.Run()
    // When you have executed the test, the Person is deleted from database
    tearDown(personID)
    os.Exit(retCode)
}

func setUp(P *Person) {
    // ...
    db.add(P)
    // ...
}

func tearDown(id int) {
    // ...
    db.delete(id)
    // ...
}

func getPerson(t *testing.T) {
    P := Get(personID)

    if P.Name != personName {
        t.Error("P.Name is %s and it must be Toni", P.Name)
    }
}
```

HTML

`go test coverprofile` `go tool`HTML。

```
    go test -coverprofile=c.out
    go tool cover -html=c.out
```

https://riptutorial.com/zh-CN/go/topic/1234/

62:

◦ ◦ ◦

- slice= make[] typelencap//
- slice = appendsliceitem//
- slice = appendsliceitems ...//
- len= lenslice//
- cap= capslice//
- elNum= copydstslice//

Examples

```
slice = append(slice, "hello", "world")
```

```
slice1 := []string{"!"}
slice2 := []string{"Hello", "world"}
slice  := append(slice1, slice2...)
```

Go Playground

/“”

;◦

slice of int◦

```
slice := []int{1, 2, 3, 4, 5, 6}
// > [1 2 3 4 5 6]
```

```
// index of first element to remove (corresponding to the '3' in the slice)
var first = 2

// index of last element to remove (corresponding to the '5' in the slice)
var last = 4
```

“”

```
// keeping elements from start to 'first element to remove' (not keeping first to remove),
// removing elements from 'first element to remove' to 'last element to remove'
// and keeping all others elements to the end of the slice
newSlice1 := append(slice[:first], slice[last+1:]...)
// > [1 2 6]

// you can do using directly numbers instead of variables
newSlice2 := append(slice[:2], slice[5:]...)
// > [1 2 6]
```

```
// Another way to do the same
newSlice3 := slice[:first + copy(slice[first:], slice[last+1:])]
// > [1 2 6]

// same that newSlice3 with hard coded indexes (without use of variables)
newSlice4 := slice[:2 + copy(slice[2:], slice[5:])]
// > [1 2 6]
```

AND

```
var indexToRemove = 3
newSlice5 := append(slice[:indexToRemove], slice[indexToRemove+1:]...)
// > [1 2 3 5 6]

// hard-coded version:
newSlice5 := append(slice[:3], slice[4:]...)
// > [1 2 3 5 6]
```

```
newSlice6 := append(slice[:0], slice[last+1:]...)
// > [6]

// That can be simplified into
newSlice6 := slice[last+1:]
// > [6]
```

```
newSlice7 := append(slice[:first], slice[first+1:len(slice)-1]...)
// > [1 2]

// That can be simplified into
newSlice7 := slice[:first]
// > [1 2]
```

last := first-1 ◦

◦ ◦

make()◦ ◦

```
var s = make([]int, 3, 5) // length 3, capacity 5
```

len()

```
var n = len(s) // n == 3
```

cap()

```
var c = cap(s) // c == 5
```

make()

```
for idx, val := range s {
    fmt.Println(idx, val)
}
```

```
// output:
// 0 0
// 1 0
// 2 0
```

play.golang.org

```
var x = s[3] // panic: runtime error: index out of range
```

```
var t = []int{3, 4}
s = append(s, t) // s is now []int{0, 0, 0, 3, 4}
n = len(s) // n == 5
c = cap(s) // c == 5
```

```
var u = []int{5, 6}
s = append(s, u) // s is now []int{0, 0, 0, 3, 4, 5, 6}
n = len(s) // n == 7
c = cap(s) // c > 5
```

◦

-

1.

```
var sourceSlice []interface{} = []interface{}{"Hello",5.10,"World",true}
```

2.

- = sourceSlice

```
var destinationSlice []interface{} = make([]interface{},len(sourceSlice))
```

3. copy

```
copy(destinationSlice,sourceSlice)
```

◦

[]Type◦

```
var a []int
```

[]Type{values}◦

```
var a []int = []int{3, 1, 4, 1, 5, 9}
```

make◦ Type lengthcapacity ◦

```
a := make([]int, 0, 5)
```

append◦

```
a = append(a, 5)
```

len◦

```
length := len(a)
```

cap◦ ◦ Go◦

```
capacity := cap(a)
```

◦

```
a[0]  // Gets the first member of `a`
```

rangefor◦ ◦

```
for index, value := range a {
    fmt.Println("Index: " + index + " Value: " + value)  // Prints "Index: 0 Value: 5" (and
continues until end of slice)
}
```

```
// Our base slice
slice := []int{ 1, 2, 3, 4 }
// Create a zero-length slice with the same underlying array
tmp := slice[:0]

for _, v := range slice {
  if v % 2 == 0 {
    // Append desired values to slice
    tmp = append(tmp, v)
  }
}

// (Optional) Reassign the slice
slice = tmp // [2, 4]
```

slice`nil` 0 ◦ nil◦ 0`[]int{}make([]int, 5)[5:]` ◦

nil`nil` slice

```
s = []int(nil)
```

```
if len(s) == 0 {
    fmt.Ptintf("s is empty.")
}
```

https://riptutorial.com/zh-CN/go/topic/733/

63:

◦ ◦

CGo◦ ◦ Go◦

Examples

```
type User struct {
    FirstName, LastName string
    Email               string
    Age                 int
}
```

◦ ◦ FirstNameLastName ◦

◦ ◦

```
type Account struct {
    UserID      int    // exported
    accessToken string // unexported
}
```

◦ ◦

```
package main

import "bank"

func main() {
    var x = &bank.Account{
        UserID: 1,          // this works fine
        accessToken: "one", // this does not work, since accessToken is unexported
    }
}
```

bankUserIdaccessToken◦

bank

```
package bank

type Account struct {
    UserID int
    accessToken string
}

func ProcessUser(u *Account) {
    u.accessToken = doSomething(u) // ProcessUser() can access u.accessToken because
                                   // it's defined in the same package
}
```

。

```
type Request struct {
    Resource string
}

type AuthenticatedRequest struct {
    Request
    Username, Password string
}
```

AuthenticatedRequest Resource Request UsernamePassword。

```
func main() {
    ar := new(AuthenticatedRequest)
    ar.Resource = "example.com/request"
    ar.Username = "bob"
    ar.Password = "P@ssw0rd"
    fmt.Printf("%#v", ar)
}
```

Request。 。 Struct。 ResourceResourcehttp://https://。 AuthenticatedRequest

```
type ResourceFormatter struct {}

func(r *ResourceFormatter) FormatHTTP(resource string) string {
    return fmt.Sprintf("http://%s", resource)
}
func(r *ResourceFormatter) FormatHTTPS(resource string) string {
    return fmt.Sprintf("https://%s", resource)
}


type AuthenticatedRequest struct {
    Request
    Username, Password string
    ResourceFormatter
}
```

```
func main() {
    ar := new(AuthenticatedRequest)
    ar.Resource = "www.example.com/request"
    ar.Username = "bob"
    ar.Password = "P@ssw0rd"

    println(ar.FormatHTTP(ar.Resource))
    println(ar.FormatHTTPS(ar.Resource))

    fmt.Printf("%#v", ar)
}
```

ResourceFormatterAuthenticatedRequest。

。 ResourceFormatterAuthenticatedRequest。

Struct

```
type User struct {
    name string
}

func (u User) Name() string {
    return u.name
}

func (u *User) SetName(newName string) {
    u.name = newName
}
```

◦ ◦ SetName()◦

```
package main

import "fmt"

type User struct {
    name string
}

func (u User) Name() string {
    return u.name
}

func (u *User) SetName(newName string) {
    u.name = newName
}

func main() {
    var me User

    me.SetName("Slim Shady")
    fmt.Println("My name is", me.Name())
}
```

```
data := struct {
    Number int
    Text   string
} {
    42,
    "Hello world!",
}
```

```
package main

import (
    "fmt"
)

func main() {
    data := struct {Number int; Text string}{42, "Hello world!"} // anonymous struct
    fmt.Printf("%+v\n", data)
}
```

◦ reflect◦

```
struct Account {
    Username      string `json:"username"`
    DisplayName   string `json:"display_name"`
    FavoriteColor string `json:"favorite_color,omitempty"`
}
```

JSONencoding/json◦

key:"value"

```
struct StructName {
    FieldName int `package1:"customdata,moredata" package2:"info"`
}
```

encoding/xmlencoding/jsonstruct◦

◦

◦

```
type T struct {
    I int
    S string
}

// initialize a struct
t := T{1, "one"}

// make struct copy
u := t // u has its field values equal to t

if u == t { // true
    fmt.Println("u and t are equal") // Prints: "u and t are equal"
}
```

't''u'◦

Ttu◦

```
fmt.Printf("t.I = %d, u.I = %d\n", t.I, u.I) // t.I = 100, u.I = 1
```

T

```
type T struct {
    I  int
    S  string
    xs []int // a slice is a reference type
}
```

◦ ◦

```
// initialize a struct
t := T{I: 1, S: "one", xs: []int{1, 2, 3}}

// make struct copy
u := t // u has its field values equal to t
```

utxs。

```
// update a slice field in u
u.xs[1] = 500

fmt.Printf("t.xs = %d, u.xs = %d\n", t.xs, u.xs)
// t.xs = [1 500 3], u.xs = [1 500 3]
```

。

```
// explicitly initialize u's slice field
u.xs = make([]int, len(t.xs))
// copy the slice values over from t
copy(u.xs, t.xs)

// updating slice value in u will not affect t
u.xs[1] = 500

fmt.Printf("t.xs = %d, u.xs = %d\n", t.xs, u.xs)
// t.xs = [1 2 3], u.xs = [1 500 3]
```

。

```
type Point struct { X, Y int }
p := Point{1, 2}
```

。 。 。

```
anim := gif.GIF{LoopCount: nframes}
```

。

。

。

```
var s struct{}
```

```
type T struct{}
```

The Go Playground

```
fmt.Println(unsafe.Sizeof(s))
```

0 ◦ The Go Playground

```
package main

import (
    "fmt"
    "time"
)

func main() {
    done := make(chan struct{})
    go func() {
        time.Sleep(1 * time.Second)
        close(done)
    }()

    fmt.Println("Wait...")
    <-done
    fmt.Println("done.")
}
```

---

https://riptutorial.com/zh-CN/go/topic/374/

64: CSV

- csvReader= csv.NewReaderr
- dataerr= csvReader.Read

Examples

CSV

CSV

```
#id,title,text
1,hello world,"This is a ""blog""."
2,second time,"My
second
entry."
```

```
// r can be any io.Reader, including a file.
csvReader := csv.NewReader(r)
// Set comment character to '#'.
csvReader.Comment = '#'
for {
    post, err := csvReader.Read()
    if err != nil {
        log.Println(err)
        // Will break on EOF.
        break
    }
    fmt.Printf("post with id %s is titled %q: %q\n", post[0], post[1], post[2])
}
```

```
post with id 1 is titled "hello world": "This is a \"blog\"."
post with id 2 is titled "second time": "My\nsecond\nentry."
2009/11/10 23:00:00 EOF
```

https //play.golang.org/p/d2E6-CGGIe 。

CSV https://riptutorial.com/zh-CN/go/topic/5818/csv

65:

Examples

Go。 。

```
package main

import (
    "fmt"
    "os"
)

func main() {

    progName := os.Args[0]
    arguments := os.Args[1:]

    fmt.Printf("Here we have program '%s' launched with following flags: ", progName)

    for _, arg := range arguments {
        fmt.Printf("%s ", arg)
    }

    fmt.Println("")
}
```

```
$ ./cmd test_arg1 test_arg2
Here we have program './cmd' launched with following flags: test_arg1 test_arg2
```

。 `os var Args []string`

Go`flag`。

`flag`GNU。 `-exampleflag` -- `--exampleflag`。 GNU。

```
package main

import (
    "flag"
    "fmt"
)

func main() {

    // basic flag can be defined like this:
    stringFlag := flag.String("string.flag", "default value", "here comes usage")
    // after that stringFlag variable will become a pointer to flag value

    // if you need to store value in variable, not pointer, than you can
    // do it like:
    var intFlag int
    flag.IntVar(&intFlag, "int.flag", 1, "usage of intFlag")
```

```
    // after all flag definitions you must call
    flag.Parse()

    // then we can access our values
    fmt.Printf("Value of stringFlag is: %s\n", *stringFlag)
    fmt.Printf("Value of intFlag is: %d\n", intFlag)

}
```

flag

```
$ ./flags -h
Usage of ./flags:
  -int.flag int
        usage of intFlag (default 1)
  -string.flag string
        here comes usage (default "default value")
```

```
$ ./flags -string.flag test -int.flag 24
Value of stringFlag is: test
Value of intFlag is: 24
```

```
$ ./flags
Value of stringFlag is: default value
Value of intFlag is: 1
```

https://riptutorial.com/zh-CN/go/topic/4023/

66:

Examples

Go`log` `Print`。

```
package main

import "log"

func main() {

    log.Println("Hello, world!")
    // Prints 'Hello, world!' on a single line

    log.Print("Hello, again! \n")
    // Prints 'Hello, again!' but doesn't break at the end without \n

    hello := "Hello, Stackers!"
    log.Printf("The type of hello is: %T \n", hello)
    // Allows you to use standard string formatting. Prints the type 'string' for %T
    // 'The type of hello is: string
}
```

io.Writer。

```
package main

import (
    "log"
    "os"
)

func main() {
    logfile, err := os.OpenFile("test.log", os.O_RDWR|os.O_CREATE|os.O_APPEND, 0666)
    if err != nil {
        log.Fatalln(err)
    }
    defer logfile.Close()

    log.SetOutput(logfile)
    log.Println("Log entry")
}
```

```
$ cat test.log
2016/07/26 07:29:05 Log entry
```

syslog

`log/syslog`syslog

```
package main
```

```
import (
    "log"
    "log/syslog"
)

func main() {
    syslogger, err := syslog.New(syslog.LOG_INFO, "syslog_example")
    if err != nil {
        log.Fatalln(err)
    }

    log.SetOutput(syslogger)
    log.Println("Log entry")
}
```

syslog

```
Jul 26 07:35:21 localhost syslog_example[18358]: 2016/07/26 07:35:21 Log entry
```

https://riptutorial.com/zh-CN/go/topic/3724/

67:

Examples

bytes.Reader

bytesio.Reader◦ Reader◦ ◦

```
message := []byte("Hello, playground")

reader := bytes.NewReader(message)

bs := make([]byte, 5)
n, err := reader.Read(bs)
if err != nil {
    log.Fatal(err)
}

fmt.Printf("Read %d bytes: %s", n, bs)
```

https://riptutorial.com/zh-CN/go/topic/7000/

68:

Examples

Go

```
// Simple type conversion
var x := Foo{}    // x is of type Foo
var y := (Bar)Foo // y is of type Bar, unless Foo cannot be cast to Bar, then compile-time
error occurs.
// Extended type conversion
var z,ok := x.(Bar)    // z is of type Bar, ok is of type bool - if conversion succeeded, z
has the same value as x and ok is true. If it failed, z has the zero value of type Bar, and ok
is false.
```

Go。type MyInterface interface {Thing}

```
type MyImplementer struct {}

func (m MyImplementer) Thing() {
    fmt.Println("Huzzah!")
}

// Interface is implemented, no error. Variable name _ causes value to be ignored.
var _ MyInterface = (*MyImplementer)nil

type MyNonImplementer struct {}

// Compile-time error - cannot case because interface is not implemented.
var _ MyInterface = (*MyNonImplementer)nil
```

Go。

```
package main

import (
    "fmt"
)

type MetersPerSecond float64
type KilometersPerHour float64

func (mps MetersPerSecond) toKilometersPerHour() KilometersPerHour {
    return KilometersPerHour(mps * 3.6)
}

func (kmh KilometersPerHour) toMetersPerSecond() MetersPerSecond {
    return MetersPerSecond(kmh / 3.6)
}

func main() {
    var mps MetersPerSecond
    mps = 12.5
    kmh := mps.toKilometersPerHour()
```

```
    mps2 := kmh.toMetersPerSecond()
    fmt.Printf("%vmps = %vkmh = %vmps\n", mps, kmh, mps2)
}
```

https://riptutorial.com/zh-CN/go/topic/2851/

69:

select◦ -◦ -◦ -◦

- {}
- {case true}
- select {case incomingData= <-someChannel}
- {}

Examples

goroutinechan◦

mainforselect◦ selectcasetrue◦ ;default◦

```
// Use of the select statement with channels (no timeouts)
package main

import (
    "fmt"
    "time"
)

// Function that is "chatty"
// Takes a single parameter a channel to send messages down
func chatter(chatChannel chan<- string) {
    // Clean up our channel when we are done.
    // The channel writer should always be the one to close a channel.
    defer close(chatChannel)

    // loop five times and die
    for i := 1; i <= 5; i++ {
        time.Sleep(2 * time.Second) // sleep for 2 seconds
        chatChannel <- fmt.Sprintf("This is pass number %d of chatter", i)
    }
}

// Our main function
func main() {
    // Create the channel
    chatChannel := make(chan string, 1)

    // start a go routine with chatter (separate, non blocking)
    go chatter(chatChannel)

    // This for loop keeps things going while the chatter is sleeping
    for {
        // select statement will block this thread until one of the two conditions below is
met
        // because we have a default, we will hit default any time the chatter isn't chatting
        select {
        // anytime the chatter chats, we'll catch it and output it
        case spam, ok := <-chatChannel:
            // Print the string from the channel, unless the channel is closed
```

```
            // and we're out of data, in which case exit.
            if ok {
                fmt.Println(spam)
            } else {
                fmt.Println("Channel closed, exiting!")
                return
            }
        default:
            // print a line, then sleep for 1 second.
            fmt.Println("Nothing happened this second.")
            time.Sleep(1 * time.Second)
        }
    }
}
```

Go Playground

select with timeouts

for3selectcase◦ selectcasechatter()◦

```
// Use of the select statement with channels, for timeouts, etc.
package main

import (
    "fmt"
    "time"
)

// Function that is "chatty"
//Takes a single parameter a channel to send messages down
func chatter(chatChannel chan<- string) {
    // loop ten times and die
    time.Sleep(5 * time.Second) // sleep for 5 seconds
    chatChannel<- fmt.Sprintf("This is pass number %d of chatter", 1)
}

// out main function
func main() {
    // Create the channel, it will be taking only strings, no need for a buffer on this
project
    chatChannel := make(chan string)
    // Clean up our channel when we are done
    defer close(chatChannel)

    // start a go routine with chatter (separate, no blocking)
    go chatter(chatChannel)

    // select statement will block this thread until one of the two conditions below is met
    // because we have a default, we will hit default any time the chatter isn't chatting
    select {
    // anytime the chatter chats, we'll catch it and output it
    case spam := <-chatChannel:
        fmt.Println(spam)
    // if the chatter takes more than 3 seconds to chat, stop waiting
    case <-time.After(3 * time.Second):
        fmt.Println("Ain't no time for that!")
    }
}
```

https://riptutorial.com/zh-CN/go/topic/3539/

70:

。。。goroutine。

- makechan int//
- makechan int5//5
- closech//“ch”
- ch < - 1 //1“ch”
- val= <-ch //“ch”
- valok= <-ch //; okbool

`make(chan struct{})`。

goroutine。。

Examples

`range`

```
func foo() chan int {
    ch := make(chan int)

    go func() {
        ch <- 1
        ch <- 2
        ch <- 3
        close(ch)

    }()

    return ch
}

func main() {
    for n := range foo() {
        fmt.Println(n)
    }

    fmt.Println("channel is now closed")
}
```

```
1
2
3
channel is now closed
```

。

```
func main() {
    // Create a buffered channel to prevent a goroutine leak. The buffer
    // ensures that the goroutine below can eventually terminate, even if
```

```
    // the timeout is met. Without the buffer, the send on the channel
    // blocks forever, waiting for a read that will never happen, and the
    // goroutine is leaked.
    ch := make(chan struct{}, 1)

    go func() {
        time.Sleep(10 * time.Second)
        ch <- struct{}{}
    }()

    select {
    case <-ch:
        // Work completed before timeout.
    case <-time.After(1 * time.Second):
        // Work was not completed after 1 second.
    }
}
```

goroutines

goroutine

```
func main() {
    ch := make(chan struct{})
    go func() {
        // Wait for main thread's signal to begin step one
        <-ch

        // Perform work
        time.Sleep(1 * time.Second)

        // Signal to main thread that step one has completed
        ch <- struct{}{}

        // Wait for main thread's signal to begin step two
        <-ch

        // Perform work
        time.Sleep(1 * time.Second)

        // Signal to main thread that work has completed
        ch <- struct{}{}
    }()

    // Notify goroutine that step one can begin
    ch <- struct{}{}

    // Wait for notification from goroutine that step one has completed
    <-ch

    // Perform some work before we notify
    // the goroutine that step two can begin
    time.Sleep(1 * time.Second)

    // Notify goroutine that step two can begin
    ch <- struct{}{}

    // Wait for notification from goroutine that step two has completed
    <-ch
```

```
}
```

```
func bufferedUnbufferedExample(buffered bool) {
    // We'll declare the channel, and we'll make it buffered or
    // unbuffered depending on the parameter `buffered` passed
    // to this function.
    var ch chan int
    if buffered {
        ch = make(chan int, 3)
    } else {
        ch = make(chan int)
    }

    // We'll start a goroutine, which will emulate a webserver
    // receiving tasks to do every 25ms.
    go func() {
        for i := 0; i < 7; i++ {
            // If the channel is buffered, then while there's an empty
            // "slot" in the channel, sending to it will not be a
            // blocking operation. If the channel is full, however, we'll
            // have to wait until a "slot" frees up.
            // If the channel is unbuffered, sending will block until
            // there's a receiver ready to take the value. This is great
            // for goroutine synchronization, not so much for queueing
            // tasks for instance in a webserver, as the request will
            // hang until the worker is ready to take our task.
            fmt.Println(">", "Sending", i, "...")
            ch <- i
            fmt.Println(">", i, "sent!")
            time.Sleep(25 * time.Millisecond)
        }
        // We'll close the channel, so that the range over channel
        // below can terminate.
        close(ch)
    }()

    for i := range ch {
        // For each task sent on the channel, we would perform some
        // task. In this case, we will assume the job is to
        // "sleep 100ms".
        fmt.Println("<", i, "received, performing 100ms job")
        time.Sleep(100 * time.Millisecond)
        fmt.Println("<", i, "job done")
    }
}
```

;◦ `fatal error: all goroutines are asleep - deadlock!`

```
package main

import "fmt"

func main() {
    msg := make(chan string)
    msg <- "Hey There"
    go func() {
        fmt.Println(<-msg)
    }()
```

```
}
```

```
package main

import "fmt"
import "time"

func main() {
    msg :=make(chan string, 1)
    msg <- "Hey There!"
    go func() {
        fmt.Println(<-msg)
    }()
    time.Sleep(time.Second * 1)
}
```

◦ sync.WaitGroup◦

```
package main

import (
    "fmt"
    "sync"
    "time"
)

func main() {
    numPiecesOfWork := 20
    numWorkers := 5

    workCh := make(chan int)
    wg := &sync.WaitGroup{}

    // Start workers
    wg.Add(numWorkers)
    for i := 0; i < numWorkers; i++ {
        go worker(workCh, wg)
    }

    // Send work
    for i := 0; i < numPiecesOfWork; i++ {
        work := i % 10 // invent some work
        workCh <- work
    }

    // Tell workers that no more work is coming
    close(workCh)

    // Wait for workers to finish
    wg.Wait()

    fmt.Println("done")
}

func worker(workCh <-chan int, wg *sync.WaitGroup) {
    defer wg.Done() // will call wg.Done() right before returning

    for work := range workCh { // will wait for work until workCh is closed
        doWork(work)
```

```
    }
}

func doWork(work int) {
    time.Sleep(time.Duration(work) * time.Millisecond)
    fmt.Println("slept for", work, "milliseconds")
}
```

https://riptutorial.com/zh-CN/go/topic/1263/

71:

Go。errnoerrnoJavatry / catch。。

。**nil**。

Go。。

error。

```
// This method doesn't fail
func DoSomethingSafe() {
}

// This method can fail
func DoSomething() (error) {
}

// This method can fail and, when it succeeds,
// it returns a string.
func DoAndReturnSomething() (string, error) {
}
```

Examples

errors。

```
errors.New("this is an error")
```

fmt

```
var f float64
fmt.Errorf("error with some additional information: %g", f)
```

```
package main

import (
    "errors"
    "fmt"
)

var ErrThreeNotFound = errors.New("error 3 is not found")

func main() {
    fmt.Println(DoSomething(1)) // succeeds! returns nil
    fmt.Println(DoSomething(2)) // returns a specific error message
    fmt.Println(DoSomething(3)) // returns an error variable
    fmt.Println(DoSomething(4)) // returns a simple error message
}

func DoSomething(someID int) error {
    switch someID {
```

```
    case 3:
        return ErrThreeNotFound
    case 2:
        return fmt.Errorf("this is an error with extra info: %d", someID)
    case 1:
        return nil
    }

    return errors.New("this is an error")
}
```

Go。 error。

```
// The error interface is represented by a single
// Error() method, that returns a string representation of the error
type error interface {
    Error() string
}
```

。

```
// Define AuthorizationError as composite literal
type AuthorizationError string

// Implement the error interface
// In this case, I simply return the underlying string
func (e AuthorizationError) Error() string {
    return string(e)
}
```

```
package main

import (
    "fmt"
)

// Define AuthorizationError as composite literal
type AuthorizationError string

// Implement the error interface
// In this case, I simply return the underlying string
func (e AuthorizationError) Error() string {
    return string(e)
}

func main() {
    fmt.Println(DoSomething(1)) // succeeds! returns nil
    fmt.Println(DoSomething(2)) // returns an error message
}

func DoSomething(someID int) error {
    if someID != 1 {
        return AuthorizationError("Action not allowed!")
    }

    // do something here
```

```
    // return a nil error if the execution succeeded
    return nil
}
```

Go。 `error`。

```
// This method can fail
func DoSomething() error {
    // functionThatReportsOK is a side-effecting function that reports its
    // state as a boolean. NOTE: this is not a good practice, so this example
    // turns the boolean value into an error. Normally, you'd rewrite this
    // function if it is under your control.
    if ok := functionThatReportsOK(); !ok {
        return errors.New("functionThatReportsSuccess returned a non-ok state")
    }

    // The method succeeded. You still have to return an error
    // to properly obey to the method signature.
    // But in this case you return a nil error.
    return nil
}
```

。

```
// This method can fail and, when it succeeds,
// it returns a string.
func DoAndReturnSomething() (string, error) {
    if os.Getenv("ERROR") == "1" {
        return "", errors.New("The method failed")
    }

    s := "Success!"

    // The method succeeded.
    return s, nil
}

result, err := DoAndReturnSomething()
if err != nil {
    panic(err)
}
```

Go。 `error`。

```
func DoAndReturnSomething() (string, error) {
    if os.Getenv("ERROR") == "1" {
        return "", errors.New("The method failed")
    }

    // The method succeeded.
    return "Success!", nil
}
```

。

```
result, err := DoAndReturnSomething()
if err != nil {
    panic(err)
}

// This is executed only if the method didn't return an error
fmt.Println(result)
```

_°

```
result, _ := DoAndReturnSomething()
fmt.Println(result)
```

。 。

。

```
func Foo() error {
    return errors.New("I failed!")
}

func Bar() (string, error) {
    err := Foo()
    if err != nil {
        return "", err
    }

    return "I succeeded", nil
}

func Baz() (string, string, error) {
    res, err := Bar()
    if err != nil {
        return "", "", err
    }

    return "Foo", "Bar", nil
}
```

“”。 。 。

```
type Foo struct {
    Is []int
}

func main() {
    fp := &Foo{}
    if err := fp.Panic(); err != nil {
        fmt.Printf("Error: %v", err)
    }
    fmt.Println("ok")
}

func (fp *Foo) Panic() (err error) {
    defer PanicRecovery(&err)
    fp.Is[0] = 5
```

```
    return nil
}

func PanicRecovery(err *error) {

    if r := recover(); r != nil {
        if _, ok := r.(runtime.Error); ok {
             //fmt.Println("Panicing")
             //panic(r)
             *err = r.(error)
        } else {
            *err = r.(error)
        }
    }
}
```

。

https://riptutorial.com/zh-CN/go/topic/785/

72:

- nil。

Examples

Go。 。

```
var myString string      // "" - an empty string
var myInt int64          // 0 - applies to all types of int and uint
var myFloat float64      // 0.0 - applies to all types of float and complex
var myBool bool          // false
var myPointer *string    // nil
var myInter interface{}  // nil
```

。 nil。 。

。

```
var myIntSlice []int    // [] - an empty slice
```

make。

```
myIntSlice := make([]int, 5)    // [0, 0, 0, 0, 0] - a slice with 5 zeroes
fmt.Println(myIntSlice[3])
// Prints 0
```

myIntSliceint50int。

new。

```
myIntSlice := new([]int)        // &[] - a pointer to an empty slice
*myIntSlice = make([]int, 5)    // [0, 0, 0, 0, 0] - a slice with 5 zeroes
fmt.Println((*myIntSlice)[3])
// Prints 0
```

myIntSlice[3](*myIntSlice)[3] 。

。

```
type ZeroStruct struct {
    myString string
    myInt    int64
    myBool   bool
}

func main() {
    var myZero = ZeroStruct{}
    fmt.Printf("Zero values are: %q, %d, %t\n", myZero.myString, myZero.myInt, myZero.myBool)
```

```
    // Prints "Zero values are: "", 0, false"
}
```

Go

;

`myIntArrayint`0

```
var myIntArray [5]int     // an array of five 0's: [0, 0, 0, 0, 0]
```

https://riptutorial.com/zh-CN/go/topic/6069/

73:

Examples

◦ Java`nil`◦ Go

◦ falsezero◦

```
func isAlive() bool {
    //Not implemented yet
    return false
}
```

false◦ /◦

Go

```
package main

import "fmt"

func isAlive() (bool, error) {
    //Not implemented yet
    return false, fmt.Errorf("Not implemented yet")
}

func main() {
    _, err := isAlive()
    if err != nil {
        fmt.Printf("ERR: %s\n", err.Error())
    }

}
```

◦

- nil◦
- `nil`

```
func(d *DB) GetUser(id uint64) (*User, error) {
    //Some error ocurred
    return nil, err
}
```

https://riptutorial.com/zh-CN/go/topic/6379/

74:

*UserUser

Examples

Go。 。 unqiue。

```
package main

type User struct {
    ID uint64
    FullName string
    Email    string
}

func main() {
    user := User{
        1,
        "Zelalem Mekonen",
        "zola.mk.27@gmail.com",
    }

    fmt.Println(user) // {1 Zelalem Mekonen zola.mk.27@gmail.com}
}
```

```
type User struct {
    ID uint64
    FullName, Email string
}

user := new(User)

user.ID = 1
user.FullName = "Zelalem Mekonen"
user.Email = "zola.mk.27@gmail.com"
```

struct。 。

```
package main

type Admin struct {
    Username, Password string
}

type User struct {
    ID uint64
    FullName, Email string
    Admin // embedded struct
}

func main() {
    admin := Admin{
        "zola",
```

```
        "supersecretpassword",
    }

    user := User{
        1,
        "Zelalem Mekonen",
        "zola.mk.27@gmail.com",
        admin,
    }

    fmt.Println(admin) // {zola supersecretpassword}

    fmt.Println(user) // {1 Zelalem Mekonen zola.mk.27@gmail.com {zola supersecretpassword}}

    fmt.Println(user.Username) // zola

    fmt.Println(user.Password) // supersecretpassword
}
```

Go

`structs function int string boolinterfaces` interface。

`structsmethodsclass`。

Go

```
func (name receiverType) methodName(paramterList) (returnList) {}
```

```
package main

type Admin struct {
    Username, Password string
}

func (admin Admin) Delete() {
    fmt.Println("Admin Deleted")
}

type User struct {
    ID uint64
    FullName, Email string
    Admin
}

func (user User) SendEmail(email string) {
    fmt.Printf("Email sent to: %s\n", user.Email)
}

func main() {
    admin := Admin{
        "zola",
        "supersecretpassword",
    }

    user := User{
        1,
        "Zelalem Mekonen",
        "zola.mk.27@gmail.com",
```

```
        admin,
    }

    user.SendEmail("Hello") // Email sent to: zola.mk.27@gmail.com

    admin.Delete() // Admin Deleted
}
```

Vs

◦ anoter◦

vaule◦

```
package main

type User struct {
    ID uint64
    FullName, Email string
}

// We do no require any special syntax to access field because receiver is a pointer
func (user *User) SendEmail(email string) {
    fmt.Printf("Sent email to: %s\n", user.Email)
}

// ChangeMail will modify the users email because the receiver type is a ponter
func (user *User) ChangeEmail(email string) {
    user.Email = email;
}

func main() {
    user := User{
        1,
        "Zelalem Mekonen",
        "zola.mk.27@gmail.com",
    }

    user.SendEmail("Hello") // Sent email to: zola.mk.27@gmail.com

    user.ChangeEmail("zola@gmail.com")

    fmt.Println(user.Email) // zola@gmail.com
}
```

◦ ◦ Go◦

◦ Go

```
package main

type Runner interface {
    Run()
}

type Admin struct {
```

```
    Username, Password string
}

func (admin Admin) Run() {
    fmt.Println("Admin ==> Run()");
}

type User struct {
    ID uint64
    FullName, Email string
}

func (user User) Run() {
    fmt.Println("User ==> Run()")
}

// RunnerExample takes any type that fullfils the Runner interface
func RunnerExample(r Runner) {
    r.Run()
}

func main() {
    admin := Admin{
        "zola",
        "supersecretpassword",
    }

    user := User{
        1,
        "Zelalem Mekonen",
        "zola.mk.27@gmail.com",
    }

    RunnerExample(admin)

    RunnerExample(user)

}
```

https://riptutorial.com/zh-CN/go/topic/8801/

75:

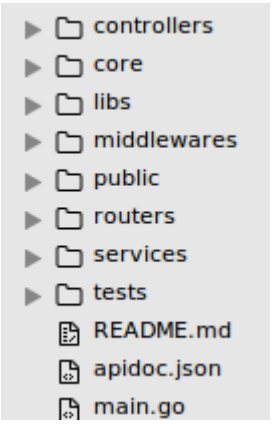
Examples

GinRestfull Projects API

GinGolangWeb。API40。。

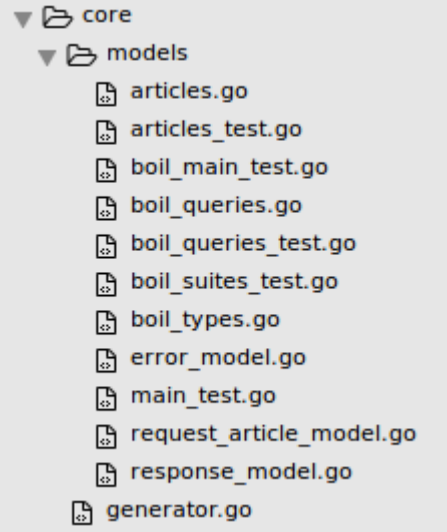
8+ main.go

1.
2.
3.
4.
5.
6.
7.
8.
9. main.go

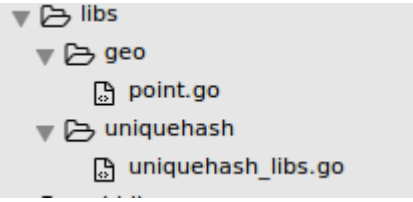


ControllersAPI。API

ORM

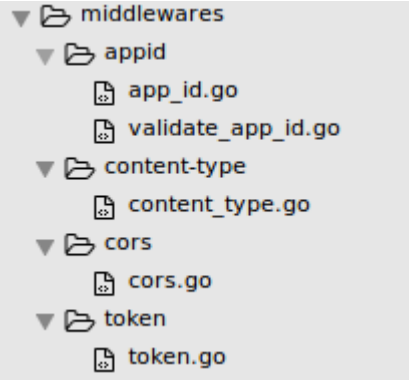


◦ /`go get package_name`◦ ◦



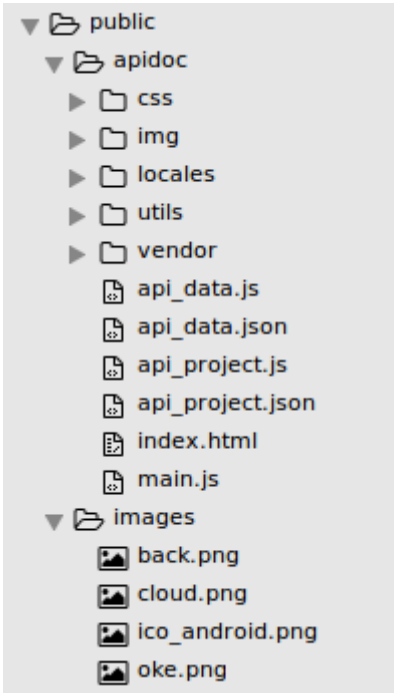
---

corsdevice-idauth/

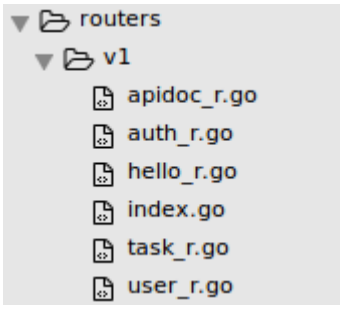


---

pacakgehtmlcssjavascript



REST API。



。

auth_r.go

```
import (
    auth "simple-api/controllers/v1/auth"
    "gopkg.in/gin-gonic/gin.v1"
)

func SetAuthRoutes(router *gin.RouterGroup) {

/**
 * @api {post} /v1/auth/login Login
 * @apiGroup Users
 * @apiHeader {application/json} Content-Type Accept application/json
 * @apiParam {String} username User username
 * @apiParam {String} password User Password
 * @apiParamExample {json} Input
 *    {
 *      "username": "your username",
 *        "password"     : "your password"
 *    }
 * @apiSuccess {Object} authenticate Response
 * @apiSuccess {Boolean} authenticate.success Status
```

```
 * @apiSuccess {Integer} authenticate.statuscode Status Code
 * @apiSuccess {String} authenticate.message Authenticate Message
 * @apiSuccess {String} authenticate.token Your JSON Token
 * @apiSuccessExample {json} Success
 *    {
 *        "authenticate": {
 *               "statuscode": 200,
 *              "success": true,
 *           "message": "Login Successfully",
 *              "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRyc

 *            }
 *      }
 * @apiErrorExample {json} List error
 *    HTTP/1.1 500 Internal Server Error
 */

    router.POST("/auth/login" , auth.Login)
}
```

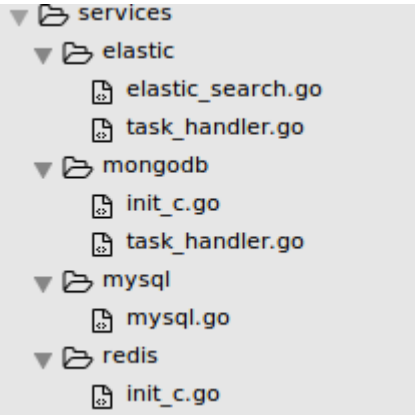
◦ APIapidoc◦ index.go

index.go

```
package v1


import (
    "gopkg.in/gin-gonic/gin.v1"
    token "simple-api/middlewares/token"
    appid "simple-api/middlewares/appid"
)
func InitRoutes(g *gin.RouterGroup)  {
    g.Use(appid.AppIDMiddleWare())
    SetHelloRoutes(g)
    SetAuthRoutes(g) // SetAuthRoutes invoked
    g.Use(token.TokenAuthMiddleWare())  //secure the API From this line to bottom with JSON
Auth
    g.Use(appid.ValidateAppIDMiddleWare())
    SetTaskRoutes(g)
    SetUserRoutes(g)
}
```

---

mongodbredismysqlelasticsearch◦



---

main.go

API。。

main.go

```
package main
import (
    "fmt"
    "net/http"
    "gopkg.in/gin-gonic/gin.v1"
    "articles/services/mysql"
    "articles/routers/v1"
    "articles/core/models"
)

var router  *gin.Engine;

func init() {
    mysql.CheckDB()
    router = gin.New();
    router.NoRoute(noRouteHandler())
    version1:=router.Group("/v1")
    v1.InitRoutes(version1)

}

func main() {
    fmt.Println("Server Running on Port: ", 9090)
    http.ListenAndServe(":9090",router)
}

func noRouteHandler() gin.HandlerFunc{
    return  func(c *gin.Context) {
    var statuscode     int
    var message        string         = "Not Found"
    var data         interface{} = nil
    var listError [] models.ErrorModel = nil
    var endpoint    string = c.Request.URL.String()
    var method        string = c.Request.Method

    var tempEr models.ErrorModel
```

```
    tempEr.ErrorCode     = 4041
    tempEr.Hints         = "Not Found !! \n Routes In Valid. You enter on invalid
Page/Endpoint"
    tempEr.Info            = "visit http://localhost:9090/v1/docs to see the available routes"
    listError             = append(listError,tempEr)
    statuscode             = 404
    responseModel := &models.ResponseModel{
        statuscode,
        message,
        data,
        listError,
        endpoint,
        method,
    }
    var content gin.H = responseModel.NewResponse();
    c.JSON(statuscode,content)
    }
}
```

ps

---

github

https://riptutorial.com/zh-CN/go/topic/9463/

S. No		Contributors
1	Go	4444, alejosocorro, Alexander, Amitay Stern, Andrej Bencic, Andrii Abramov, burfl, Burhan Ali, cat, Cody Gustafson, Community, David G., Dmitri Goldring, Feckmore, Florian Hämmerle, Franck Dernoncourt, Gerep, Greg Bray, hellyale, Hunter, James Taylor, Jared Hooper, Jon Chan, Katamaritaco, Mark Henderson, Matt, mbb, MegaTom, mmlb, mnoronha, mohan08p, Nir, nix, nouney, patterns, Pavel Nikolov, ProfNandaa, Quentin Skousen, Radouane ROUFID, Rahul Nair, RamenChef, raulsntos, Sam Whited, seriousdev, Simone Carletti, skunkmb, sztanpet, Tanmay Garg, Topo, Unapiedra, Vikash, Xavier Nicollet
2	Base64	Nathan Osman, RamenChef, Sam Whited
3	CGO	MaC, Vojtech Kane
4	FMT	Lanzafame, Nevermore, Sam Whited
5	GoJWT	AniSkywalker
6	GoProtobuf	mohan08p
7	Go	ganesh kumar, Harshal Sheth, Ingve, Lanzafame, Mayank Patel , Nevermore, Quentin Skousen, Sam Whited, theflametrooper, Vikash
8	HTTP	1lann, dmportella, Lanzafame, Sam Whited, SommerEngineering
9	HTTP	Chief, frigo americain, Jon Erickson, Kin, Nathan Osman, rogerdpack, Sam Whited, Sascha, seriousdev, Simone Carletti, SommerEngineering, Tanmay Garg, Zhinkk
10	IOTA	4444, Florian Hämmerle, Ingve, mohan08p, Sam Whited, Wojciech Kazior, Zoyd
11	JSON	Dmitry Udod, Joe, Jon Chan, Kyle Brandt, Nathan Osman, RamenChef, Sam Whited, shayan, Simone Carletti, sztanpet, Tanmay Garg, Utahcon
12	SQL	Adrian, artamonovdev, bernardn, Francesco Pasa, Nevermore, Sam Whited, Sascha, Tanmay Garg, wrfly
13	Vendoring	Abhilekh Singh, Boris Le Méec, burfl, Dmitri Goldring, Ivan

		Mikushin, Mark Henderson, Martin Campbell, Michael, Sam Whited, Vardius
14	XML	ivarg, Sam Whited
15	YAML	Nathan Osman, Orr, Sam Whited
16		Ingaz, Sam Whited
17		Ankit Deshpande
18		Ainar-G, NatNgs, raulsntos
19		Adrian, Prutswonder
20		Jordan, Katamaritaco, mbb, mohan08p, RamenChef, Riley Guerin, SH', Siu Ching Pong -Asuka Kenji-, SommerEngineering, sztanpet, Zoyd
21	AtomGo	Ali M, Danny Chen, Katamaritaco
22	gopprof	mbb, Nevermore, radbrawler
23		abhink
24		Elijah Sarver, Grzegorz Żur, Kenny Grant
25		Sam Whited
26		zola
27		burfl, Community, ganesh kumar, Ingve, nk2ge5k
28		ecem
29		Boris Le Méec, Dmytro Sadovnychyi, Grzegorz Żur, jayantS, LeoTao, Nathan Osman, nouney, palestamp, RamenChef, Right leg, Thomas Gerot
30		SommerEngineering
31		dmportella, Grzegorz Żur, icza, Michael, Nathan Osman, RadicalFish, RamenChef, skunkmb, tkausl
32		ganesh kumar, mammothbane, radbrawler
33	/	Utahcon
34		Community, FredMaggiowski, Jon Chan, Simone Carletti
35		putu

36		Abhay, abhink, Amitay Stern, Brendan, burfl, chowey, Chris Lucas, cizixs, Community, creker, Dair, Dmitri Goldring, gbulmer , Hugo, James, JepZ, Joe, Kaedys, Kamil Kisiel, Kyle Brandt, Mark Henderson, matt.s, Milo Christiansen, NatNgs, Oleg Sklyar, radbrawler, RamenChef, Roland Illig, Sam Whited, seh, Simone Carletti, skunkmb, Surreal Dreams, Vojtech Kane, Zoyd , Zyerah
37		mohan08p
38		sadlil
39		burfl, photoionized, seriousdev
40		Pavel Nikolov, RamenChef, Sam Whited, Simone Carletti
41		Chris Lucas, Community, Florian Hämmerle, flyingfinger, Grzegorz Żur, Harshal Sheth, Ilya, Inanc Gumus, Kyle Brandt, Nathan Osman, Roland Illig, Ryan Kelln, Tim S. Van Haren, VonC, zianwar, Zoyd
42		4444, RamenChef, Sam Whited, seriousdev
43		1lann, burfl, Community, ivan73, jayantS, Jon Chan, mgh, MohamedAlaa, RamenChef, Sam Whited, Steven Maude, Thomas Gerot
44		JunLe Meng, Kaedys, Kristoffer Sall-Storgaard, Sam Whited
45		Krzysztof Kowalczyk, Kyle Brandt, Nevermore
46		David Hoelzer, Jon Chan, Joost, Mal Curtis, metmirr, Nevermore, skunkmb
47		Cody Roseborough, dotctor, Francis Norton, Grzegorz Żur, icza, Ingve, meysam, Mike, ptman, sadlil, Sam Whited, Wendy Adi
48	I / O.	Abhilekh Singh
49		abhink, Adrian, Sam Whited, Vikash
50		Sam Whited
51		Community, Sam Whited, Utahcon
52		NatNgs, nouney, Noval Agung Prayogo, Sam Whited
53	I / O.	1lann, Andres Kütt, greatwolf, Grzegorz Żur, koblas, noisewaterphd, Quentin Skousen, Sam Whited

54	+ HTML	Stephen Rudolph
55		ganesh kumar, Pavel Kazhevets
56		Lanzafame, NatNgs, raulsntos
57		Pavel Kazhevets, RamenChef, Tanmay Garg
58		Florian Hämmerle, Sourabh
59		Adrian, Ankit Deshpande, Harshal Sheth, ivan.sim, Jared Ririe, Nathan Osman, Omid, Pavel Nikolov, Rodolfo Carvalho, seriousdev, Toni Villena, Zoyd
60		1lann, Benjamin Kadish, burfl, cizixs, Grzegorz Żur, Guillaume, Jared Hooper, Joost, Jukurrpa, Kyle Brandt, Mark Henderson, NatNgs, RamenChef, Simone Carletti, skunkmb, Tanmay Garg, Zoyd
61		abhink, Amitay Stern, Anthony Atkinson, Blixt, burfl, cizixs, Community, FredMaggiowski, Howl, Ingve, Kin, MaC, Mark Henderson, matt.s, mohan08p, Nathan Osman, nouney, Patrick, Quentin Skousen, radbrawler, RamenChef, Roland Illig, Simone Carletti, sunkuet02, Vojtech Kane, Wojciech Kazior
62	CSV	Ainar-G
63		Ingve, Pavel Kazhevets, Sam Whited
64		Grzegorz Żur, Jon Chan, Nathan Osman, Pavel Kazhevets, Sam Whited
65		Mike Houston
66		Adrian, Florian Hämmerle
67		Harshal Sheth, Kaedys, RamenChef, Sam Whited, Utahcon
68		Chris Lucas, Howl, Jeremy, Kwarrtz, metmirr, RamenChef, Rodolfo Carvalho, Zoyd
69		bowsersenior, elevine, Elijah Sarver, Florian Hämmerle, groob, Ingve, Joe, Kin, Paul Hankin, Quentin Skousen, Sam Whited, Simone Carletti, Sridhar, Surreal Dreams, Vervious, Zoyd
70		Harshal Sheth, raulsntos, Surreal Dreams
71		Davyd Dzhahaiev, Sam Whited, zola
72		Iman Tumorang