image

Go 103

Pallat Anchaleechamaikorn

yod.pallat@gmail.com
https://github.com/pallat
https://dev.to/pallat

https://go.dev/tour (Thai)
https://github.com/uber-go/guide (Thai)

image

Composition with Struct Embedding

```go
type Card struct {
	HolderName string
	IssuedAt  time.Time
	ExpiredAt time.Time
}

type DrivingCard struct {
	DriverLicense   string
	Class           string
	Card
}

type IDCard struct {
	ID      string
	Address string
	Card
}
```

image

Composition (2)

```go
type Card struct {
    HolderName string
    IssuedAt  time.Time
    ExpiredAt time.Time
}

func (c Card) IsExpire() bool {
    return time.Now().After(c.ExpiredAt)
}

type DrivingCard struct {
    Card
}

dc := DrivingCard{}
dc.IsExpire()
```

image

Composition (3)

```go
type ReadWriter interface {
    Reader
    Writer
}
```

image

Generic

```go
func min(x, y float64) float64 {
  if x < y {
    return x
  }
  return y
}
```

image

Generic: type parameter

```go
func min[T constraints.Ordered](x, y T) T {
  if x < y {
    return x
  }
  return y
}
```

> instantiation

```go
m := min[int](2, 3)
fmin := min[float64]
m := fmin(2.1, 2.0)
```

Parameter Type

```
type Tree[T interface{}] struct {
  left, right *Tree[T]
  data        T
}

func (t *Tree[T]) Lookup(x T) *Tree[T]

var stringTree Tree[string]
```

Type constraint

```go
interface {
  int|string|bool
}
```

```go
package constraints

type Ordered interface {
  Integer|Float|~string
}
```

image

First-Class Function

```go
var add = func(a, b int) int {
    return a + b
}

fmt.Println(add(1, 2))
```

image

Higher-Order Function

```
func hof(fn func(string) string) {
    ...
}

func hof() func(string) string {
    ...
}
```

image

Higher-Order Function Blog

https://dev.to/pallat/hof-in-go-18mm

image

Closure Function

```
func main() {
    fn1, fn2 := factory()
    fn1()
    fn1()
    fmt.Println(fn2())

    fn1()
    fmt.Println(fn2())
}

func factory() (func(), func() int) {
    var i int
    return func() {
            i++
        },
        func() int {
            return i
        }
}
```

image

func type

```go
type IntnFunc func(int) int
```

image

method on function

```go
type IntnFunc func(int) int

func (fn IntnFunc) Intn(n int) int {
    return fn(n)
}
```

image

Demo: RandomSay with IntnFunc

```go
var intn IntnFunc = r.Intn
```

image

goroutine

```go
func main() {
    total := 10
    now := time.Now()
    for i := 0; i < total; i++ {
        go printout(i)
    }
    fmt.Println(time.Now().Sub(now))
}

func printout(i int) {
    fmt.Println(i)
}
```

image

goroutine waiting

```go
var wg = sync.WaitGroup{}

func main() {
    total := 10
    wg.Add(total)
    now := time.Now()
    for i := 0; i < total; i++ {
        go printout(i)
    }
    wg.Wait()
    fmt.Println(time.Now().Sub(now))
}

func printout(i int) {
    fmt.Println(i)
    wg.Done()
}
```

image

channel

keyword `chan`

- no buffered channel
- buffered channel

image

buffered channel

```go
total := 10
ch := make(chan int, total)
for i := total; i > 0; i-- {
    ch <- i
}
close(ch)

for i := range ch {
    fmt.Println(i)
}
```

image

no buffered channel

```go
func main() {
    total := 10
    ch := make(chan struct{})
    now := time.Now()
    for i := 0; i < total; i++ {
        go printout(i, ch)
    }
    for i := 0; i < total; i++ {
        <-ch
    }
    fmt.Println(time.Now().Sub(now))
}

func printout(i int, ch chan struct{}) {
    fmt.Println(i)
    ch <- struct{}{}
}
```

image

Excercise - Count down 1 min

2 goroutine

1. print `+` every 1 sec
2. print `-` every 5 sec

program end after 1 minute pass

image

Goroutine exercise

```
import "github.com/pallat/force"

force.Decrypt
force.Validate()
```

brute force all encrypted

image