DEVHINTS.IO

Edit

Go cheatsheet

Hello world

hello.go

```
package main

import "fmt"

func main() {
  message := greetMe("world")
  fmt.Println(message)
}

func greetMe(name string) string {
  return "Hello, " + name + "!"
}
```

```
$ go build
```

Or try it out in the Go repl, or A Tour of Go.

Variables

Variable declaration

```
var msg string
msg = "Hello"
```

Shortcut of above (Infers type)

```
msg := "Hello"
```

Basic types

Strings

```
str := "Hello"
```

```
str := `Multiline
string`
```

Numbers

Typical types

```
num := 3          // int
num := 3.         // float64
num := 3 + 4i     // complex128
num := byte('a')  // byte (alias for
```

Other types

Strings are of type `string`.

```
var u uint = 7        // uint (unsign
var p float32 = 22.7  // 32-bit float
```

Pointers

```
func main () {

  fmt.Println("Value is", b)
}
```

```
func getPointer () (myPointer *int) {
  a := 234

}
```

Pointers point to a memory location of a variable. Go is fully garbage-collected.

See: Pointers

Type conversions

```
i := 2
f := float64(i)
u := uint(i)
```

See: Type conversions

Flow control

Conditional

```
  rest()

  groan()

  work()
}
```

See: If

Statements in if

```
  fmt.Println("Uh oh")
}
```

A condition in an `if` statement can be prece

See: If with a short statement

For-Range loop

```
entry := []string{"Jack","John","Jones"}
for i, val := range entry {
  fmt.Printf("At position %d, the character %s is present\n", i, val)
}
```

See: For-Range loops

For loop

```
for count := 0; count <= 10; count+
un
```

Functions

Lambdas

```
  return x > 10000
}
```

Functions are first class objects.

Multiple return types

```
a, b := getMessage()
```

```
func getMessage() (a string, b string
```

```
}
```

Packages

Importing

```
import "fmt"
import "math/rand"
```

```
import (
  "fmt"        // gives fmt.Println
  "math/rand"  // gives rand.Intn
)
```

Both are the same.

See: Importing

Aliases

```
r.Intn()
```

Packages

```
package hello
```

Every package file has to start with package

Concurrency

Goroutines

Buffered channels

```
func main() {
  // A "channel"


  // Start concurrent routines


  // Read 3 results
  // (Since our goroutines are concurrent,
  // the order isn't guaranteed!)
  fmt.Println(<-ch, <-ch, <-ch)
}
```

```
ch <- 1
ch <- 2
ch <- 3
// fatal error:
// all goroutines are asleep - deadlo
```

Buffered channels limit the amount of mess

See: Buffered channels

```
func push(name string, ch chan string) {
  msg := "Hey, " + name

}
```

Channels are concurrency-safe communication objects, used in goroutines.

See: Goroutines, Channels

Error control

Defer

```
func main() {

  fmt.Println("Working...")
}
```

Defers running a function until the surrounding function returns. The arguments are evaluated immediately, but the function call is not ran until later.

See: Defer, panic and recover

Deferring f

```
func main(


  fmt.Prir
}
```

Lambdas ar

Structs

Defining

```
func main() {
  v := Vertex{1, 2}
  v.X = 4
  fmt.Println(v.X, v.Y)
}
```

See: Structs

Literals

```
v := Vertex{X: 1, Y: 2}
```

```
// Field names can be omitted
v := Vertex{1, 2}
```

```
// Y is implicit
v := Vertex{X: 1}
```

You can also put field names.

Methods

Receivers

```
type Vertex struct {
  X, Y float64
}
```

```
  return math.Sqrt(v.X * v.X + v.Y * v.Y)
}
```

```
v: = Vertex{1, 2}
v.Abs()
```

There are no classes, but you can define functions with receivers.

See: Methods

Mutation

```
  v.X = v.
  v.Y = v.
}
```

```
v := Verte
v.Scale(0.
// `v` is
```

By defining

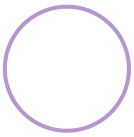
See: Pointe

References

A tour of Go (tour.golang.org)

- Golang wiki (github.com)
- Awesome Go (awesome-go.com)
- Go by Example (gobyexample.com)
- Effective Go (golang.org)
- JustForFunc Youtube (youtube.com)
- Style Guide (github.com)

▼ **6 Comments** for this cheatsheet. Write yours!



Over 381 curated cheatsheets, by developers for developers.

Devhints home

Other C-like cheatsheets

C Preprocessor
cheatsheet

C# 7
cheatsheet

Top cheatsheets

Elixir
cheatsheet

ES2015+
cheatsheet

React.js
cheatsheet

Vimdiff
cheatsheet

Vim
cheatsheet

Vim scripting
cheatsheet