Cheatography

Go Lang Cheat Sheet

by Gordon Mickel (guiltylemon) via cheatography.com/20161/cs/3006/

Deklarationen

```
var foo int // Ohne Initialisierung
var foo int = 42 // Mit Initia - lis ierung
var foo, bar int = 42, 1302 // Mehrere
var foo = 42 // Typinf erenz
foo := 42 // Kurzform, immer implizit
const constant = "This is a consta nt"
```

Loops

```
for i := 1; i < 10; i++ {...}
       for ; i < 10; {...} // while - loop
       for i < 10 {...} // Semikolons nicht nötig
       for {...} // while (true)
```

If

```
if x > 0 {
                return x
       } else {
                return -x
       }
       if a := b + c; a < 42 {
                return a
       } else {
                return a - 42
       }
```

Arrays

```
var a [10]int // int array mit 10 Elementen
a[3] = 42 // setzen
i := a[3] // auslesen
// Deklar ieren
a := [2]int{1, 2}
a := [...]i nt{1, 2} // Kompil - ierer findet die Anzahl Elemente automa tisch
```

Structs

Structs (cont)

```
>       name: "Ben",
   }
   //Zugriff auf structs
   for _, man := range []Man {man1, man2} {
         println(man.name)
   }
}
```

Slices

```
var a []int // Slice deklarieren, ähnlich wie Array aber ohne Länge
var a = []int {1, 2, 3, 4} // Slice deklar ieren und initia - lis ieren
a := []int{ 1, 2, 3, 4 } // shorthand
var b = a[lo:hi] // slice kreiieren (view of array) von index lo bis hi-1
var b = a[1:4] // slice von index 1 to 3
var b = a[:3] // ohne unterer index impliziert 0
var b = a[3:] // ohne oberer index impliziert len(a)
// Mit make Slices kreiieren
a = make([ ]byte, 5, 5) // 1. Parame nter: Länge, 2. Parameter: Kapazität =
make([ ]byte, 5) // Kapazität ist optional
```

Funktionen

Coroutines

```
import(
       t " tim e"
       f " fmt "
)
func say(s string) {
       for i := 0; i < 5; i++ {
                t.S lee p(100 * t.Mill ise cond)
                f.P rin tln(s)
       }
}
func main() {
       go say("wo rld ")
        say ("he llo ")
}
```

Channels

```
func seq(num int, c chan int) {
       x, y := 0, 1
       for num != 0 {
                x, y = y, x + y
                 num--
       }
       c <-x
}
func main() {
        //c hannels müssen vor ihrer Anwendung deklariert werden
       ch := make(chan int)
       go seq(50, ch)
       go seq(20, ch)
        res1, res2 := <-ch, <-ch
        pri ntl n(res1, res2)
//1258 6269025 6765
}
```

```
type Man struct {
	name string
	age int
}
func main() {
	man1 := Man{"Ar iel ",
26}
	man2 := Man {
		age: 24,
```

```
func functionName() {}
func functi onN ame (param1
string, param2 int) {}
// parameter mit dem gleichen
Typ
func functi onN ame (pa ram1,
param2 int) {}
// Rückga bewert
func functi onN ame() (int) {
	return 42
}
// Mehrere Rückga bewerte
func return Multi() (int,
string) {
	return 42, " foo bar "
}
```



By **Gordon Mickel**
(guiltylemon)

cheatography.com/guiltylemon/

Not published yet.
Last updated 12th May, 2016.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com