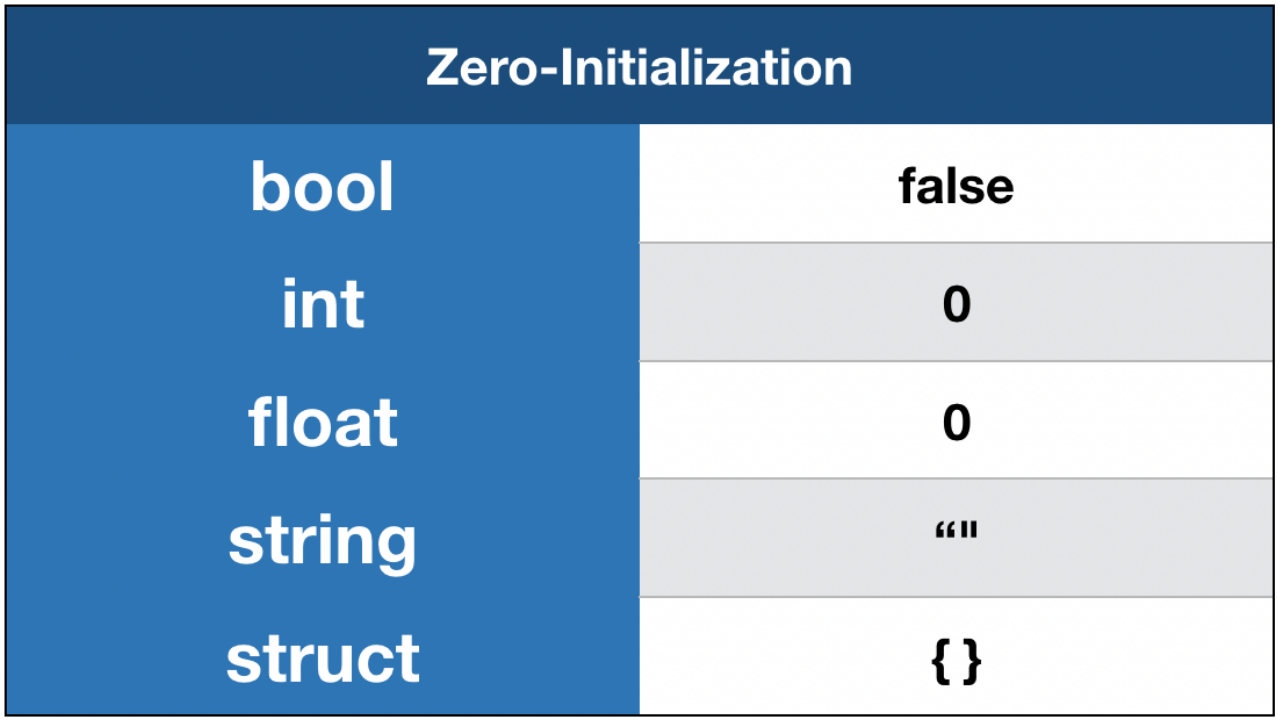
06

Para saber mais

Inicialização zero e nil

Mesmo não provendo nenhum valor, o Go garante inicializar todas as variáveis, conforme a imagem abaixo:



Porém, em muitas linguagens existe uma maneira de denotar um ponteiro nulo que, essencialmente, não aponta para nenhum lugar. Por exemplo: em C é `NULL`, em Python é `None` e em java é `null`. Em Go, temos o `nil`.

Nil e Inferência

Observe o seguinte exemplo:

```
package main

import (
    "fmt"
)

func main() {
    a := nil
    fmt.Println(a)
}
```

Será que vai compilar?

Não, não vai. O compilador imprimirá o seguinte erro: *use of untyped nil*, que significa uso não tipado do nil.

Aqui estamos tentando atribuir um valor nil apontando para algum lugar sem fornecer seu tipo e esperamos que o compilador deduza isso. O compilador não sabe se esta variável é um inteiro, uma string, um array ou uma structure.

Neste link, você pode executar o código acima. (https://play.golang.org/p/kVcneO6nVdS)

Nil com um tipo definido

Sabendo disso, observe o exemplo abaixo onde apontamos para um tipo definido:

```
package main

import (
    "fmt"
)

func main() {
    var s *string = nil
    fmt.Println(s)
}
```

Neste caso, o programa compila e retorna `<nil>` como esperado.

Neste link, você pode executar o código acima. (https://play.golang.org/p/gkwKo7rholt)

Comparando zero value

Para finalizar, observe o seguinte programa, no qual criamos duas variáveis: uma `float64` e uma `int`, sem atribuir valor, e as comparamos:

```
package main

import (
    "fmt"
)

func main() {
    var f float64
    var i int

    fmt.Println(f==i)
}
```

Neste link, você pode executar o código acima. (https://play.golang.org/p/Ri6dLuyhjeg)

Conclusão

Recebemos uma mensagem com um erro informando que os tipos são incompatíveis. Não podemos comparar o valor atribuído pela inicialização zero se temos tipos diferentes. Portanto, por mais que o Go garanta a inicialização zero de diferentes tipos, devemos ficar atentos com os tipos que estamos trabalhando.