

4 - Organizing Data with Structs

Structs in Go

Card

"Ace of Spades"

What's the suit?

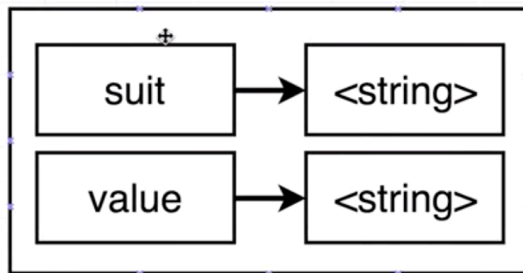
What's the value?

Struct

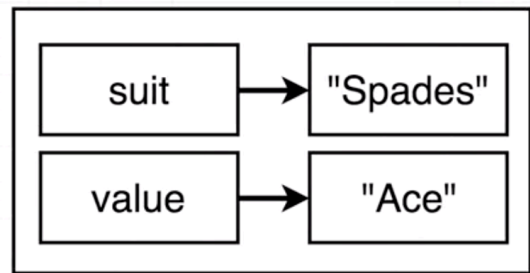
Struct

Data structure. Collection of properties that are related together.

Card Struct Field Definition

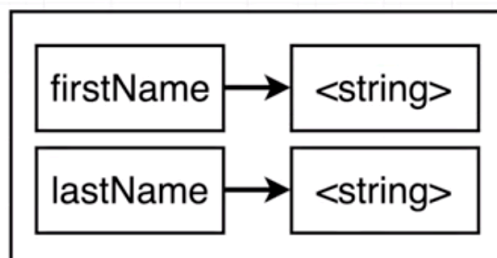


Card Struct



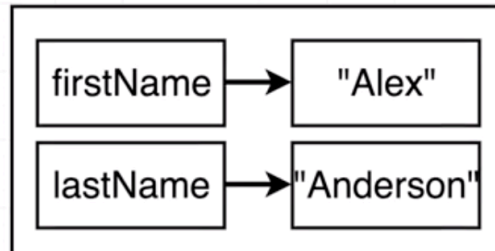
Defining Structs

Tell go what fields the person struct has



person struct

Create a new value of type person



/src/struct/main.go

```
1 type person struct {
2     firstName string
3     lastName  string
4 }
```

Declaring Structs

/src/struct/main.go

```
1 func main() {
2     alex := person{firstName: "Alex", lastName: "Anderson"}
3     fmt.Println(alex)
4 }
```

Updating Struct Values

Type	Zero Value
string	""
int	0
float	0
bool	false

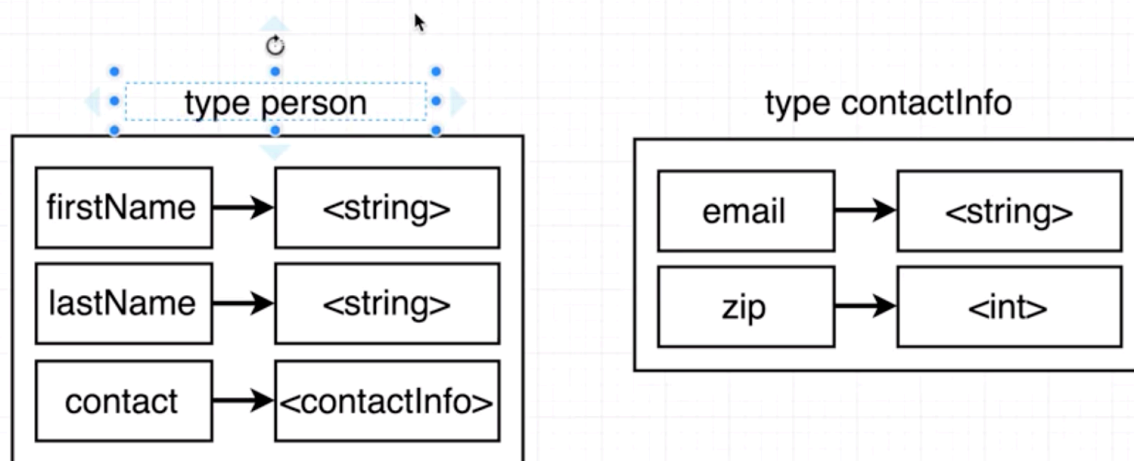
/src/struct/main.go

```

1 func main() {
2     var alex person
3
4     alex.firstName = "Alex"
5     alex.lastName = "Anderson"
6
7     fmt.Println(alex)
8     fmt.Printf("%+v", alex)
9 }

```

Embedding Structs



/src/struct/main.go

```
1 type contactInfo struct {
2     email    string
3     zipCode  int
4 }
5
6 type person struct {
7     firstName string
8     lastName  string
9     contact   contactInfo
10 }
11
12 func main() {
13     mer := person{
14         firstName: "Mer",
15         lastName:   "JQ",
16         contact: contactInfo{
17             email:    "mer@gmail.com",
18             zipCode: 94000,
19         },
20     }
21
22     fmt.Println(mer)
23 }
```

Structs with Receiver Functions

src/struct/main.go

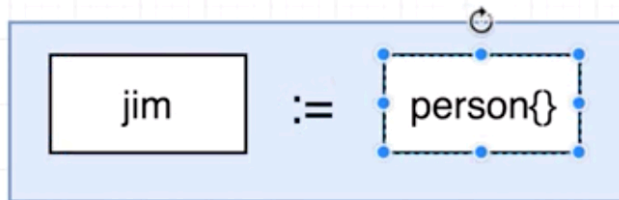
```
1 type contactInfo struct {
2     email    string
3     zipCode  int
4 }
5
6 type person struct {
7     firstName string
8     lastName  string
9     contactInfo
10 }
11
12 func main() {
13     jim := person{
14         firstName: "Jim",
15         lastName:   "Party",
16         contactInfo: contactInfo{
17             email:    "jim@gmail.com",
18             zipCode: 94000,
19         },
20     }
21
22     fmt.Println(jim)
23 }
```

Add receiver functions

/src/struct/main.go

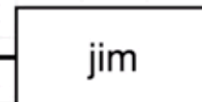
```
1 func main() {  
2     mer.updateName("Jieqiong")  
3     mer.print()  
4 }  
5  
6 func (p person) print() {  
7     fmt.Println(p)  
8 }  
9  
10 func (p person) updateName(newFirstName string) {  
11     p.firstName = newFirstName  
12 }
```

Pass By Value



RAM

Address	Value
0000	
0001	person{firstName: "Jim"....}
0002	
0003	
0004	



```
jim.updateName("Jimmy")
```

```
func (p person) updateName() {
```

RAM

Address	Value
0000	
0001	person{firstName: "Jim"....}
0002	
0003	person{firstName: "Jim"....}
0004	

Diagram illustrating memory layout (RAM) and variable pointers:

- Variable `jim` points to memory address 0001.
- Variable `p` points to memory address 0003.
- Memory addresses 0001 and 0003 contain a `person` struct: `person{firstName: "Jim"....}`.
- A dashed box highlights the memory region from address 0002 to 0004, indicating a contiguous block of memory.

Structs with Pointers

/src/struct/main.go

```
1 func main() {  
2     merPointer := &mer  
3     merPointer.updateName("Jieqiong")  
4     mer.print()  
5 }  
6  
7 func (p *person) updateName(newFirstName string) {  
8     (*p).firstName = newFirstName  
9 }
```

Pointer Operations

&variable

Give me the memory address of the value this variable is pointing at

*pointer

Give me the value this memory address is pointing at

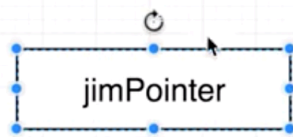
`&variable`

Give me the memory address of the value this variable is pointing at

`*pointer`

Give me the value this memory address is pointing at

`jimPointer := &jim`

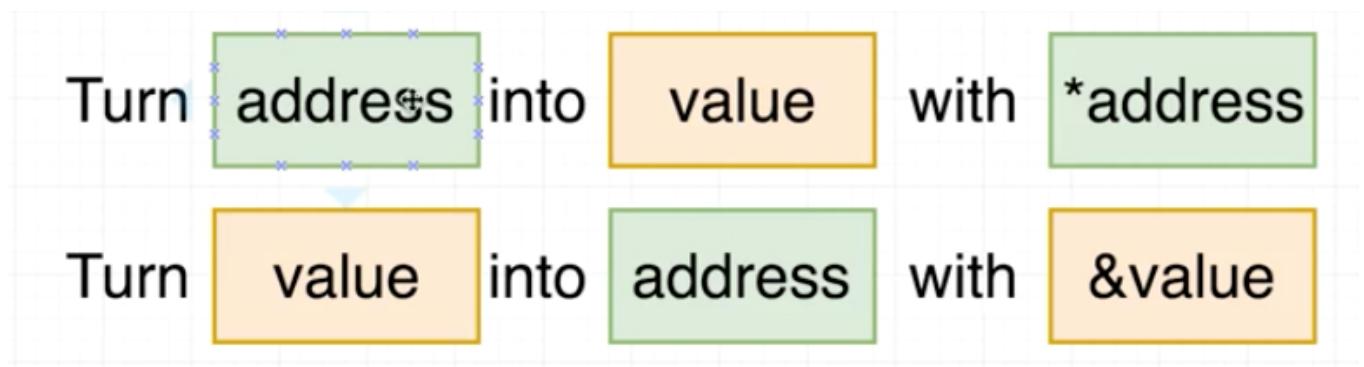
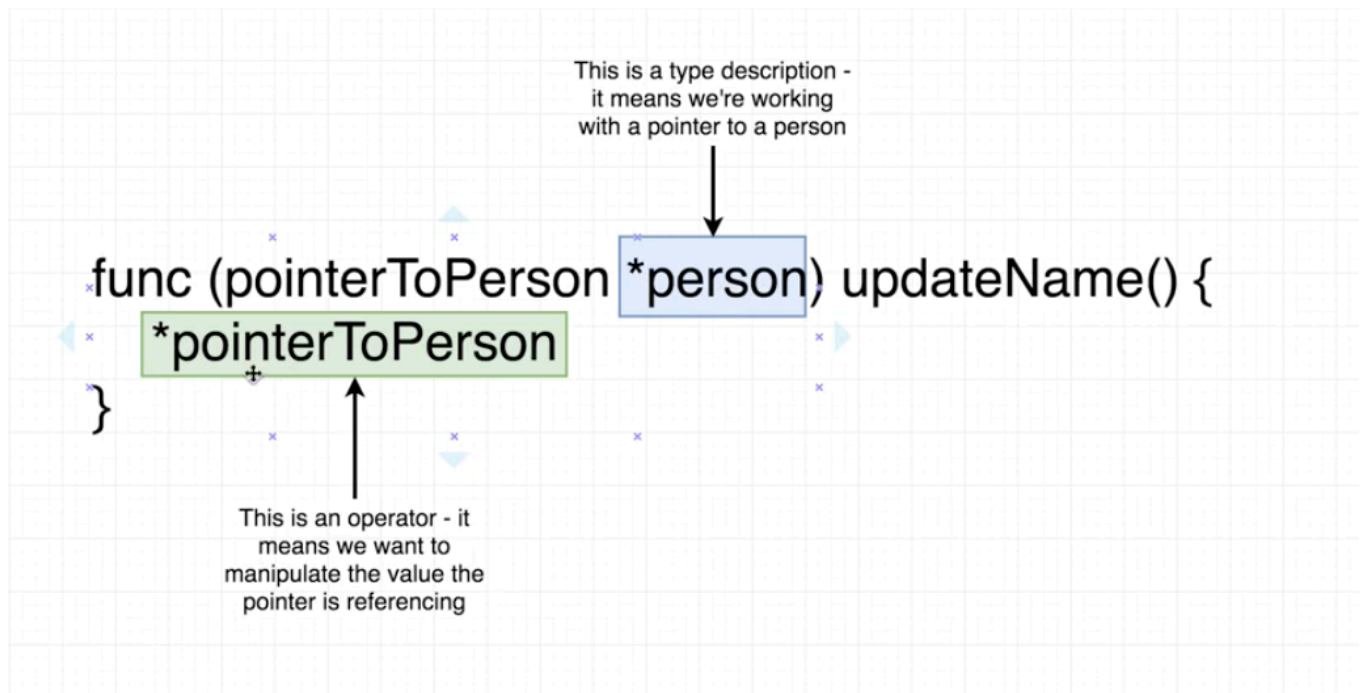


RAM

Address	Value
0000	
0001	person{firstName: "Jim"....}
0002	
0003	
0004	

`jim`

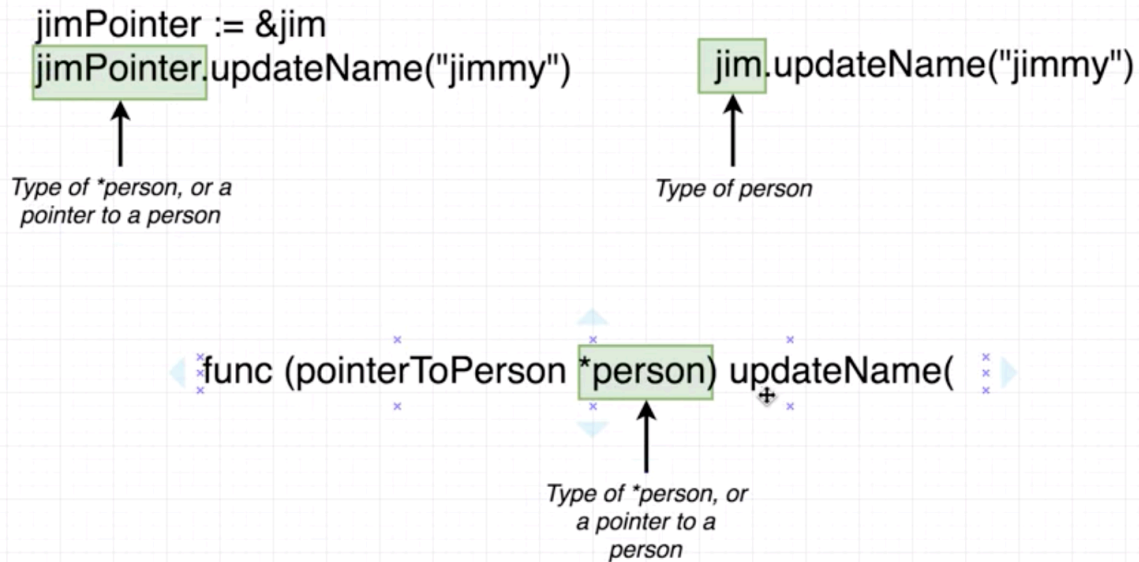




Pointer Shortcut

main.go

```
1 func main() {
2     mer.updateName("Jieqiong")
3     mer.print()
4 }
5
6 func (pointerToPerson *person) updateName(newFirstName string) {
7     (*pointerToPerson).firstName = newFirstName
8 }
```

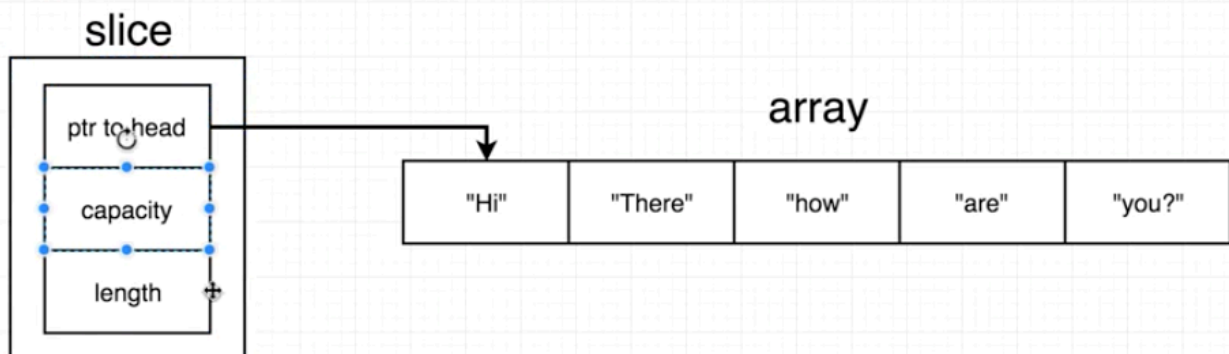
Gotchas With Pointers

/src/struct/main.go

```
1 func main() {
2     mySlice := []string{"Hi", "There", "How", "Are", "You"}
3     updateSlice(mySlice)
4     fmt.Println(mySlice)
5 }
6
7 func updateSlice(s []string) {
8     s[0] = "Bye"
9 }
```

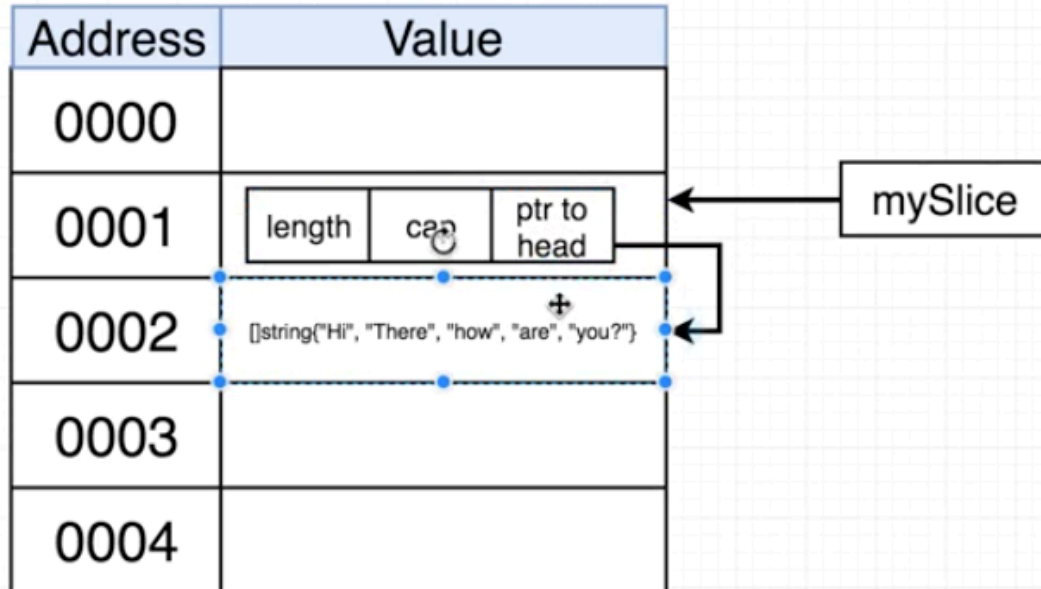
Reference vs Value Types

```
mySlice := []string{"Hi", "There", "how", "are", "you?"}
```

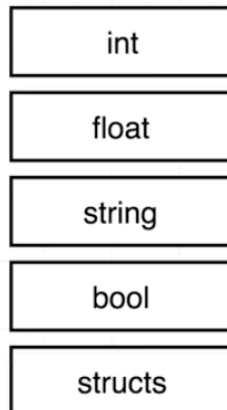


```
mySlice := []string{"Hi", "There", "how", "are", "you?"}
```

RAM

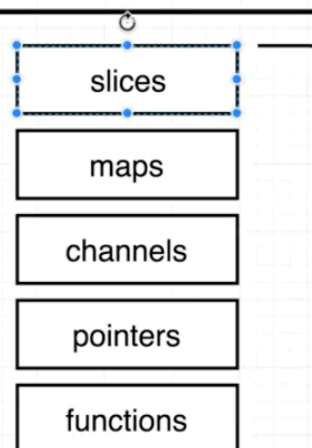


Value Types



Use pointers to
change these things
in a function

Reference Types



Don't worry about
pointers with these

