Working with Maps

Nigel Poulton
Author & Trainer

@nigelpoulton nigelpoulton.com

Maps

Key-sorted unless with a range loop

Maps

Key-value pairs (dictionary, hash table...)

<key> : <value>

Sunderland : 6

Man Utd : 20

Liverpool : 19

Chelsea : 6

Newcastle : 4

<key> : <value>

Sunderland : 6

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 6

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 7

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 8

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 9

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 10

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 11

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

<key> : <value>

Sunderland : 12

Man Utd : 20

Liverpool : 19

Chelsea : 6

Man City : 7

Agenda

- Getting started with maps
- Iterating maps
- Updating maps
- Misc
- Recap

Getting Started with Maps

```
map[<key-type>]<value-type>
```

Key type needs to be a comparable type (works with == and !=)

Keys must be unique

Misc.

Go maps

Reference types

Cheap to pass around

Specify size

(for larger maps)

Not thread-safe

Recap

Go maps recap

Declaring maps

```
map[string]float64
```

Map ordering

Unordered when retrieved with range loop

Maps are dynamic

```
myMap[<key>] = <value>
delete(myMap, <key>)
```

Maps are reference types

Passed to functions by reference

Up Next:
Working with Structs