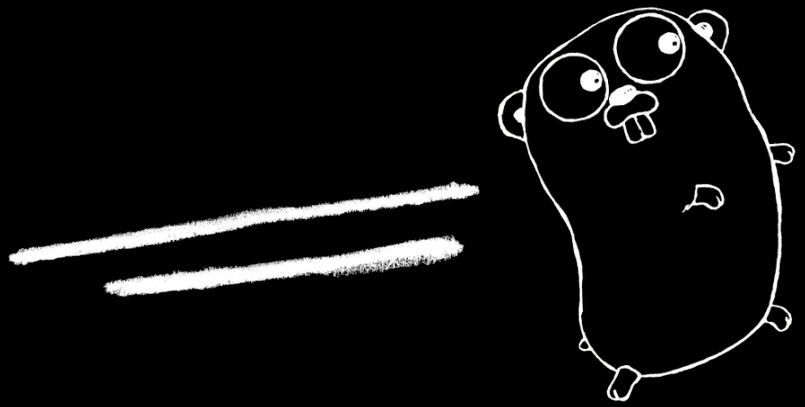


Eine Einführung in go

Hauke Stieler

16. Januar 2018

Fachbereich Informatik der Universität Hamburg

Agenda

Thank to Fred, on whose slides I was able to create these :)

Agenda

- some history
- basic features
- cool web stuff
- concurrency
- interfaces

Why go?

In 2007, three guys at Google were frustrated with the existing languages for writing server software:

- Compiling C++ was too slow
- Writing Java felt too verbose
- Aversion against inheritance and design patterns
- Getting concurrency right was hard

C++

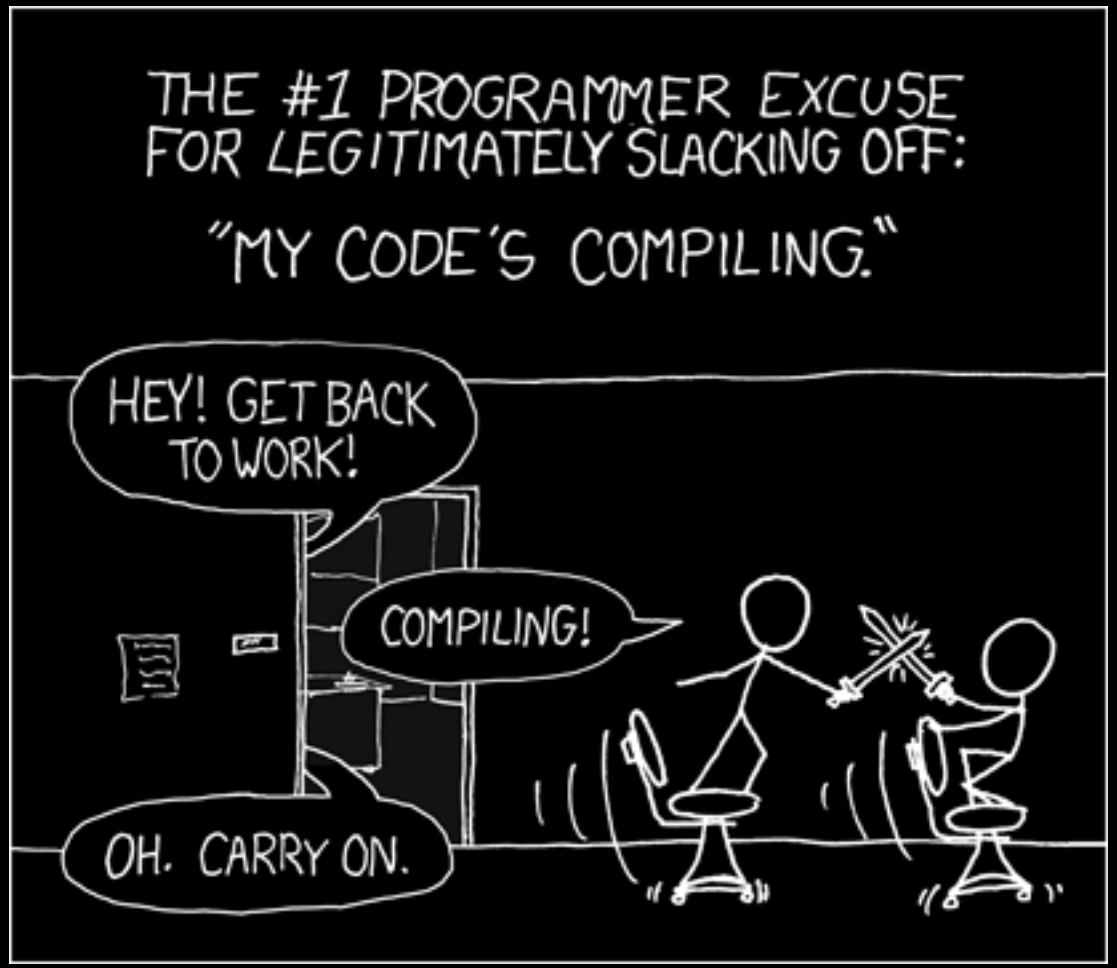
```
1  // Within large projects, popular header files
2  // get included thousands of times and hence
3  // have to be recompiled over and over again
4  #include <iostream>
5  #include <string>
6  #include <vector>
```

C++

```
1  // Within large projects, popular header files
2  // get included thousands of times and hence
3  // have to be recompiled over and over again
4  #include <iostream>
5  #include <string>
6  #include <vector>
```

gcc copies specified file by `#include` recursively into source file. The *same* header file gets recompiled over and over again.

→ Rob Pike: Public Static Void at OSCON 2010


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

Java

Let's do some Java.
Write a `public class Person` that does the following:

- store a string `name`
- store an int age

Simple, right?

Java

Let's do some Java.
Write a `public class Person` that does the following:

- store a string `name`
- store an int `age`

Simple, right?

NO :(

```java
public class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

```

Java II

```
18      public int getAge() {
19          return age;
20      }
21
22      public void setAge(int age) {
23          this.age = age;
24      }
25
26      @Override
27      public String toString() {
28          return "Person [" + "name=" + name + ", "
            ↪  +     "age=" + age + "]";
29      }
30
31      @Override
32      public int hashCode() {
33          final int prime = 31;
```

Java III

```
34          int result = 1;
35          result = prime * result + age;
36          result = prime * result + ((name == null) ? 0 :
             ↪  name.hashCode());
37          return result;
38      }
39
40      @Override
41      public boolean equals(Object obj) {
42          if (this == obj)
43              return true;
44          if (obj == null)
45              return false;
46          if (getClass() != obj.getClass())
47              return false;
48          Person other = (Person) obj;
49          if (age != other.age)
```

Java IV

```
50              return false;
51          if (name == null) {
52              if (other.name != null)
53                  return false;
54          } else if (!name.equals(other.name))
55              return false;
56          return true;
57      }
58  }
```

Design

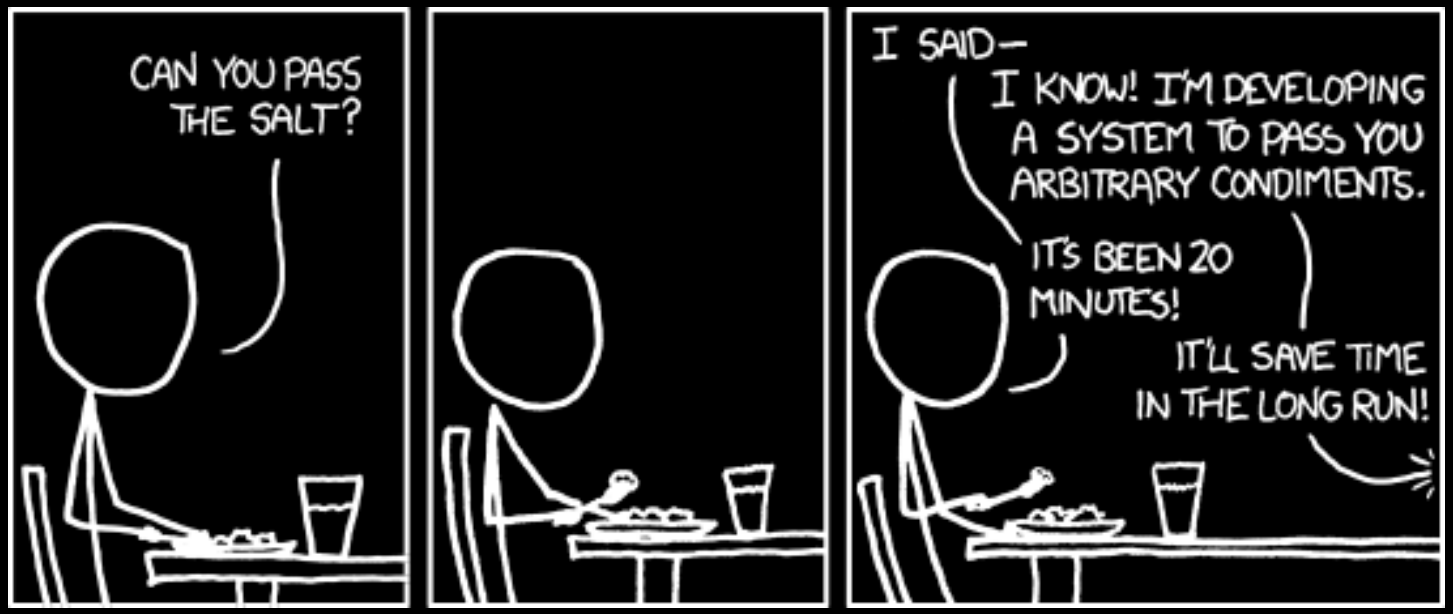
Initial design by 3 people with different backgrounds:

- Rob Pike (Concurrency)
- Robert Griesemer (Modules)
- Ken Thompson (Operating Systems)

All design decisions had to be agreed upon unanimously. Design team later joined by more people at Google.

Design

- **simplicity**
- **simplicity**
- **simplicity**
- clean package model for fast compilation
- built-in concurrency based on CSP
- interfaces instead of inheritance
- no radical changes after Go 1.0


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

Hello world!

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("hello world")
7  }
```

- The import declaration imports entire packages
- All imported names must be qualified
- Uppercase names are visible to other packages
- Unused imports are compile-time errors!

Hello world!

Get the go compiler:

```
$ sudo apt-get install golang-go
$ sudo pacman -S go

...or download from https://golang.org/dl
```

Run the code[1]:

```
$ go run hello.go
```

[1] The `go run` command works for single files, not always for projects

Basics

Keywords:

```
break       default     func        interface   select
case        defer       go          map         struct
chan        else        goto        package     switch
const       fallthrough if          range       type
continue    for         import      return      var
```

Constants:

```
true        false       nil         iota
```

Basics

Functions:

```
new         len         complex     panic
make        cap         real        recover
close       append      imag
            copy
            delete
```

Basics

Basic types:

```
int     int8     int16     int32     int64
uint    uint8    uint16    uint32    uint64    uintptr

float32      float64
complex64    complex128

bool    byte     rune      string    error
```

- `int` and `uint` are platform-dependent
- `byte` is the same as `uint8`
- `rune` is the same as `uint32`
- `uintptr` is large enough to hold pointers
- `error` is a special type for error handling

Basics

Operators:

```
*    /    %    &    &^   <<   >>
+    -    ^    |
==   !=   <    <=   >    >=
&&
||
```

- only 5 precedence levels!
- `^` is both bitwise-xor (infix) and bitwise-not (prefix)
- `&^` is bitwise-andn

Declarations

```
// three semantically identical alternatives
var x int = 0
var x int
var x = 0

// fourth alternative for local variables only
x := 0
```