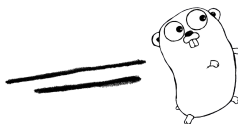


# *Die Programmiersprache Go - Eine Einführung*

## **Seminarvortrag**

Student: Adrian Helberg  
Prüfer: Prof. Dr. Axel Schmolitzky

23. März 2018



## Geschichte

Entwickler

Entwurfsphase

Veröffentlichung

Go-Community

## Merkmale

Einsatzbereich

## Syntax

## Paradigmen

## Zentrale Fragestellung

## Pro & Contra

## Compiler

## Einzelnachweise

*“Go is an open source programming language that makes it easy to build simple, reliable and efficient software.”*

(Go Website: [golang.org](https://golang.org))

*Go ist eine Open-Source-Programmiersprache, die es einfach macht, einfache, zuverlässige und effiziente Software zu erstellen.*

(Eigene Übersetzung)

# Entwickler

- ▶ Konzipiert September 2007
- ▶ Robert Griesemer, Rob Pike und Ken Thompson
- ▶ Mitarbeiter von Google LLC ®
- ▶ Aus Frust heraus entstanden
- ▶ *“Complexity is multiplicative”* - Rob Pike

# Entwurfsphase

- ▶ Ausdrucksstarke und effiziente Kombination aus Kompilierung und Ausführung
- ▶ Ähnlichkeiten mit C
- ▶ Adaptiert gute Ideen aus einigen Programmiersprachen: Pascal, Modula-2, Oberon, Oberon-2, Alef, ...

# Entwurfsphase

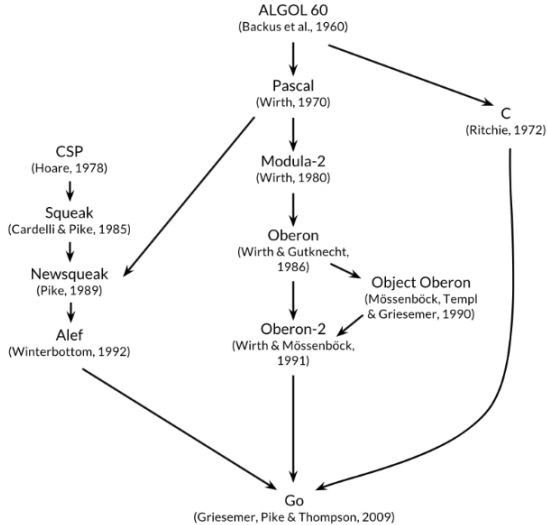


Abbildung: The Go Programming Language, Preface xii

# Entwurfsphase

- ▶ Vermeiden von features, die zu komplexen, unzuverlässigen code führen würden
- ▶ Möglichkeiten zur Nebenläufigkeit sind neu und effizient
- ▶ Datenabstraktion und Objektorientierung sind ungewohnt flexibel
- ▶ Automatische Speicherverwaltung (garbage collection)

- ▶ Vorgestellt November 2009
- ▶ Berühmt als Nachfolger für nicht typisierte Skriptsprachen  
→ Verbindung aus Ausdruckskraft und Sicherheit



# Go-Community

- ▶ Open-source projekt
  - Quellcode des Compilers, Bibliotheken (libraries) und Tools sind frei verfügbar
- ▶ Aktive, weltweite Community
- ▶ Läuft auf Unix, Mac und Windows
  - Üblicherweise ohne Modifikation transpotrierbar

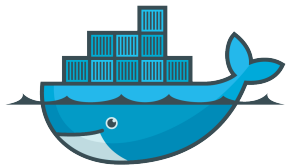
# Einsatzbereich

- ▶ “Alles was auf dem Server läuft”
- ▶ Ökosystem
- ▶ vorhandene Libraries und Projekte

- ▶ Reflection
- ▶ Typsicherheit
- ▶ Automatische Speicherbereinigung
- ▶ Interfaces, Mixins → Objektorientierung
- ▶ Keine Klassen → Java Vergleich
- ▶ Pakete

# Projekte

*Go*



docker

*Java*



elasticsearch

# Syntax

## Go

```
package main

import "fmt"

func main() {
    fmt.Println("Hello World!")
}
```

## Java

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

# Paradigmen

- ▶ Nebenläufig
- ▶ Imperativ
- ▶ Strukturiert
- ▶ Modular
- ▶ Objektorientiert

# Zentrale Fragestellung

„flexibel wie dynamisch getyped,  
aber mit statischer Typsicherheit?“

# Pro & Contra

## Pro

- ▶ Minimalismus
- ▶ Statisches Duck-Typing
- ▶ Parallelisierung
- ▶ Aufgeräumte Syntax
- ▶ Schneller Compiler

## Contra

- ▶ Keine generische Programmierung
- ▶ nil statt Option
- ▶ Wenig grundlegende Datenstrukturen
- ▶ Keine Methodenüberladung



# Pro & Contra

## Pro

- ▶ Statisch gelinkte Binärdateien
- ▶ Laufzeiteigenschaften
- ▶ Integriertes Unit-Test-Framework
- ▶ Paketmanager

## Contra

- ▶ Unbefriedigende API-Dokumentation
- ▶ Teilweise umständliche APIs
- ▶ Umständliches Mocking
- ▶ Kleines Ökosystem

# Compiler

- ▶ Gc
- ▶ Gccgo

# Einzelnachweise

