```
fun kotlin() = true
```

Kevin Böckler

9. Januar 2025

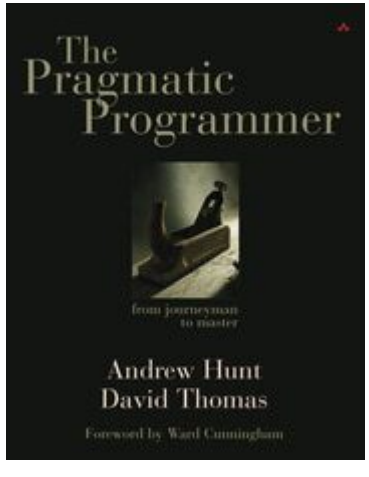
https://www.meetup.com/softwerkskammer-luebeck

AGENDA

1. Motivation
2. Wie?
3. Warum?
4. Kotlin

MOTIVATION

Learn a language every year



MOTIVATION

Besser streamen

```
findAllUsers()
    .filter { it.activated }
    .map { it.name }
    .take(3)
    .joinToString(",")
    .let { println(it) }
```

MOTIVATION

Moderne Sprache

```
fun someInts(): List<Int> {
    val list: MutableList<Int> = mutableListOf(1, 2)
    list.add(3)
    return list
}
```

WIE?

Läuft auf der JVM
-> Also auch standalone wie Java

- Kotlin Compiler als Java-Library
- Buildsystem: Maven, Gradle
- Buildartefakte: .class, .jar, .war (Java Bytecode)

WIE?

Qualitätsmerkmale nach ISO 25010

Funktionale Eignung

System bietet Funktionalität, die den angegebenen und implizierten Bedürfnissen entspricht

Zuverlässigkeit

System führt Funktionen unter den festgelegten Umgebungen aus

Sicherheit

System schützt Informationen und Daten

Wartbarkeit

System kann modifiziert werden, um es zu verbessern, zu korrigieren oder an Änderungen anzupassen

Übertragbarkeit

System kann auf verschiedenen Umgebungen betrieben werden

Benutzbarkeit

System kann von festgelegten Benutzern verwendet werden, um vorgegebene Ziele zu erreichen

Kompatibilität

System kann Informationen mit anderen Systemen austauschen

Leistungseffizienz

System liefert angemessene Geschwindigkeit mit den bereitgestellten Ressourcen

Übertragbarkeit ✓ Sicherheit ✓ Kompatibilität ✓ Portierbarkeit ✓

WARUM?

- Wartbarkeit ✓
- Nah an der Java-Sprache
- Moderner Sprachfeatures
- Aufgeräumte Syntax
- Interoperabilität

SYNTAX

```kotlin
fun main() {
    println("Hello from Kotlin")
}
```

```java
public class Main {

  public static void main() {
    System.out.println("Hello from Java");
  }
}
```

SYNTAX

```
val zahl = 12
val andereZahl: Double

var veraenderlich = 1
veraenderlich = 2

fun getSomeValue(input: String): Double {}
```

```
final var zahl = 12;
final double andereZahl;

var veraenderlich = 1;
veraenderlich = 2;

double getSomeValue(String input) {}
```

NULL-SAFETY

```
val name: String = "Kevin"
val nameOrNot: String? = "Kevin"
val nameOrNot: String? = null
```

```
String name = "Kevin";
Optional<String> nameOrNot = Optional.of("Kevin");
Optional<String> nameOrNot = Optional.empty();
```

NULL-SAFETY

```
val length = nameOrNot.length
```

```
if (nameOrNot != null) {
    val length = nameOrNot.length
}
val forcedLength = nameOrNot!!.length
```

```
if (nameOrNot.isPresent()) {
  int length = nameOrNot.get().length();
}
int forcedLength = nameOrNot.get().length();
```

NULL-SAFETY

```
// Person besteht aus Name?
val vorname = person.name?.vorname
```

```
// Person besteht aus Optional von Name
var vorname = person.getName().map(Name::vorname);
```

NULL-SAFETY

```
person.name?.let { personName ->
    println(personName)
}
```

```
person.getName().ifPresent(
    personName -> {
      System.out.println(personName);
    });
```

NULL-SAFETY

```
val myName: String = nameOrNot ?: "Max Mustermann"
```

```
var myName = nameOrNot.orElse("Max");
```

TYPES

```
class SomeClass(zahl: Int) {

    constructor() : this(1)

    private val zahl: Int = zahl

    fun printIt() {
        println(zahl)
    }
}
```

- class
- data class
- enum class

TYPES

```kotlin
data class SomeDataClass(val zahl: Int)
```

```java
class SomeOldDataClass {
    private final int zahl;

    private SomeOldDataClass(int zahl) {
      this.zahl = zahl;
    }

    public int getZahl() {
      return zahl;
    }

    @Override
    public boolean equals(Object o) {
      if (this == o) return true;
      if (o == null || getClass() != o.getClass()) return fals
```

```java
record SomeDataClass(int zahl) {}
```

TYPES

```
interface SomeInterface {
    fun printIt()
}

class SomeImplementation : SomeInterface {
    override fun printIt() {
        println("Hallo from Implementation")
    }
}
```

```
interface SomeInterface {
    void printIt();
}

class SomeImplementation implements SomeInterface {
    @Override
    public void printIt() {
      System.out.println("Hallo from Implementation");
    }
}
```

TYPES

https://www.youtube.com/watch?v=KvehHqnEXuc

FUNCTIONS

Function in class

```
val obj = SomeObject("Kevin")
val objString = obj.greet()

class SomeObject(private val innerString: String) {
    fun greet(): String {
        return "Hallo $innerString";
    }
}
```

FUNCTIONS

Extension Function

```
val name = "Kevin"
val objMessage = name.greet()

fun String.greet(): String {
    return "Hallo $this"
}
```

FUNCTIONS

```
val firstFunction: (String) -> String
val upperCaseFunction: (String) -> String
firstFunction = ::takeFirstChar
upperCaseFunction = ::upperCaseChar

val result = upperCaseFunction(firstFunction("Apfel"))
```

FUNCTIONS

```
val numberFunction: (Int) -> Int

fun doWith(number: Int, numberFunction: (Int) -> Int) {
    val result = numberFunction.invoke(number)
    println(result)
}
```

FUNCTIONS

```
fun doWith(number: Int, numberFunction: (Int) -> Int) {
    ...
}

doWith(2, SquareFuncClass())
doWith(2, ::cubeIt)

doWith(2, { it - 1 })
doWith(2) { it - 1 }
```

FUNCTIONS

https://www.youtube.com/watch?v=eXT1wglHAQo

STREAMS

Funktional mit Funktionen und Lambdas arbeiten

FEATURES

Interoperabilität

FEATURES

Operatoren

https://kotlinlang.org/docs/operator-overloading.html

FEATURES

Properties

FEATURES

Datenstrukturen

FEATURES

Desctructuring

FEATURES

Zeitmessung

FEATURES

https://www.youtube.com/watch?v=OFWMtmqocV8

```kotlin
val (sprache, gelernt) = "kotlin" to true
```

Kevin Böckler

9. Januar 2025

https://www.meetup.com/softwerkskammer-luebeck