WE ARE EUDRES EUROPEAN UNIVERSITY

fh st.pölten
UNIVERSITY OF APPLIED SCIENCES

Kotlin Multiplatform mit Compose

David Grünberger

Kotlin

Was ist Kotlin?

- Programmiersprache
- Entwickelt von JetBrains (Stable-Version 2016)
- Für Android Entwicklung unterstützt seit 2017
  - "preferred Language" seit 2021
- Vollständig Open Source
- Wachsende Community
  - Erkennbar in Stackoverflow Surveys
- Backed by JetBrains & Google

Einsatzbereiche

- Android
- Server-Backend
  - Ktor: Webframework speziell für Kotlin
  - Spring Boot
- Web-Entwicklung
  - Kotlin/JS
- Desktop
  - JavaFX
- **Multiplatform**

Warum Kotlin (statt Java)?

Warum Kotlin (statt Java)?

- Einfachere aber mächtigere Syntax
  - eliminiert Boilerplate-Code (unnötige Codeteile)
  - kurz und prägnant, leserlich
- Null-Safety
  - Vermeidung von NullPointerExceptions
- Einfachere Wartbarkeit in großen Projekten
- Vollständig interoperabel mit Java
  - Kotlin-Code wird zu Java-Bytecode kompiliert

Automatische getter/setter

java

```java
public class User {
    private String name;

    public String getName() {
        return name;
    }

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

kotlin

```kotlin
class User(var name: String)
```

Null-Safety

java

```java
String name = null;
System.out.println(name.length()); // NullPointerException!
```

kotlin

```kotlin
var name: String? = null
println(name?.length ?: 0) // Sichere Alternative, gibt 0 zurück
```

Typinferenz

java

```java
int number = 42;
String text = "Hello";
```

kotlin

```kotlin
val number = 42  // Typ: Int
val text = "Hello"  // Typ: String
```

Default-Parameter

java

```java
public void greet(String name) {
    greet(name, "Hello");
}

public void greet(String name, String greeting) {
    System.out.println(greeting + ", " + name);
}
```

kotlin

```kotlin
fun greet(name: String, greeting: String = "Hello") {
    println("$greeting, $name")
}

greet("John") // Ausgabe: Hello, John
```

Extension Functions

kotlin

```kotlin
fun String.isPalindrome(): Boolean {
    return this == this.reversed()
}

println("madam".isPalindrome()) // Ausgabe: true
```

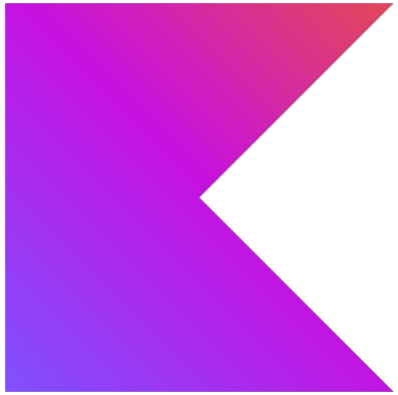
Lambdas & High-Order Functions

java

```java
List<Integer> numbers = Arrays.asList(1, 2, 3, 4);
List<Integer> squared = numbers.stream()
    .map(x -> x * x)
    .collect(Collectors.toList());
```

kotlin

```kotlin
val numbers = listOf(1, 2, 3, 4)
val squared = numbers.map { it * it }
```



Kotlin Multiplatform

Was ist Kotlin Multiplatform?

- Native Cross-Platform Framework
  - Veröffentlichung 2017
- Entwickelt von JetBrains
- Android, iOS, Web, Desktop
- Flexibel einsetzbar
  - nur geteilte Logik
  - auch geteilte UI-Komponenten möglich (Compose)
- expect/actual Mechanismus für plattformspezifischen Code

Compose

Was ist Compose?

- Deklaratives UI-Toolkit
  - Keine Trennung zwischen Layout und Logik
- Teil des Jetpack-Toolkits auf Android (Jetpack Compose)
  - state-of-the-art Methode zum Erstellen von Uis
  - hat View-System abgelöst (seperate XML Layout Files)
- Verwendbar in Kotlin Multiplatform
  - Compose Multiplatform – erweiterte plattformübergreifende Version
  - entwickelt von JetBrains

Composables

- Bausteine der UI
- Funktionen, die mit **@Composable** annotiert sind

```
@Composable
fun WelcomeMessage(name: String) {
    Text(text = "Welcome, $name!")
}
```

State Changes

- Composables reagieren auf Änderungen von States
  - Werden bei einer Änderung re-composed

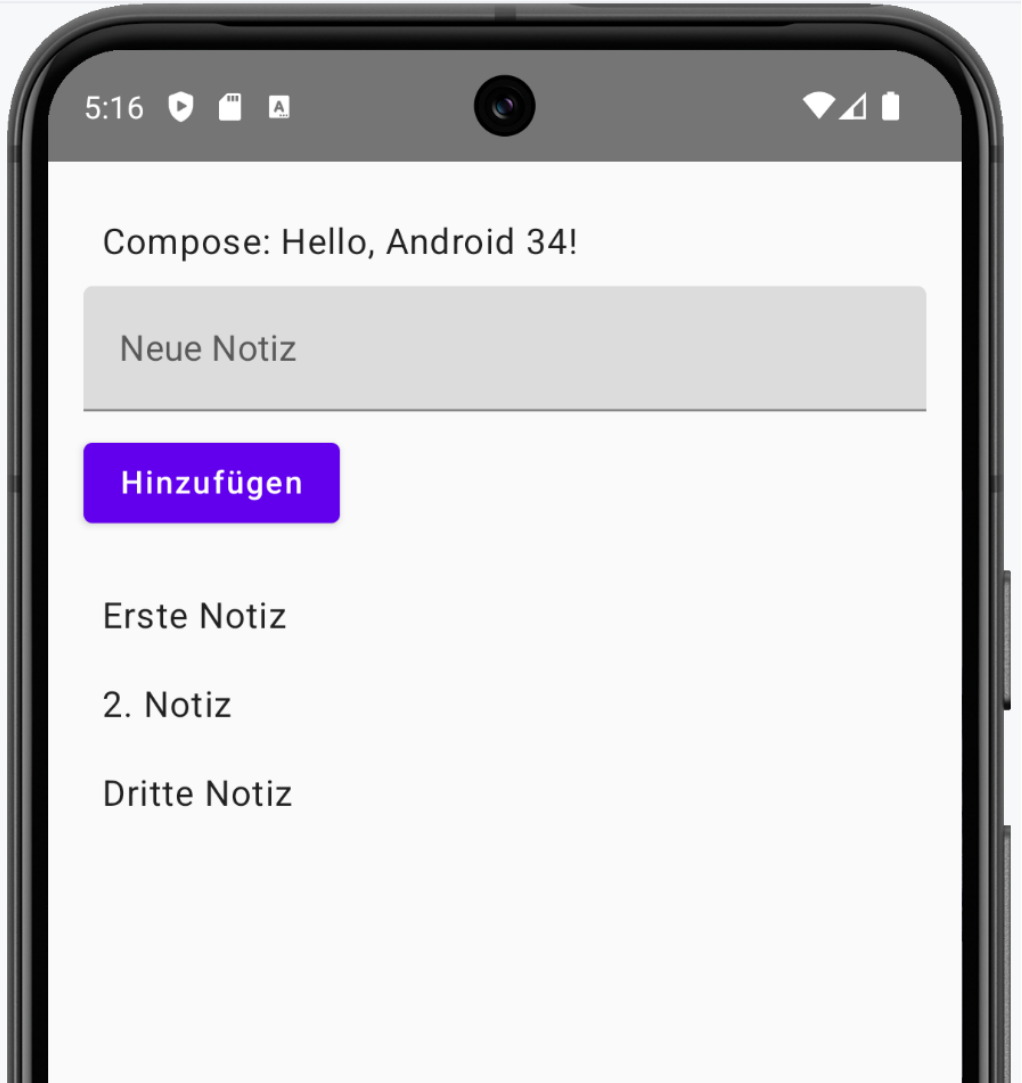
```
@Composable
fun Counter() {
    var count by remember { mutableStateOf(0) }
    Button(onClick = { count++ }) {
        Text("Count: $count")
    }
}
```

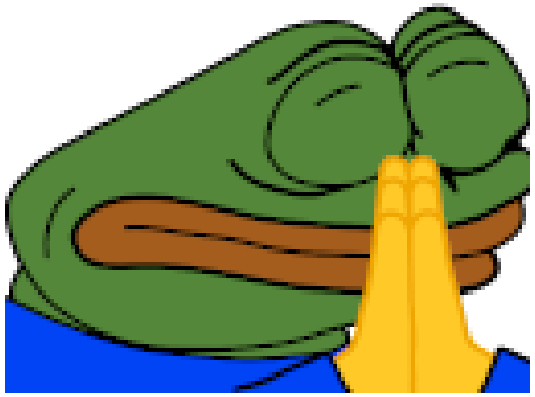
Layouts

- Compose bietet flexible Layouts wie Column, Row, Box
- Können verschachtelt werden

```
@Composable
fun LayoutExample() {
    Column {
        Text("Item 1")
        Text("Item 2")
    }
}
```

Time for coding!





Thank you!