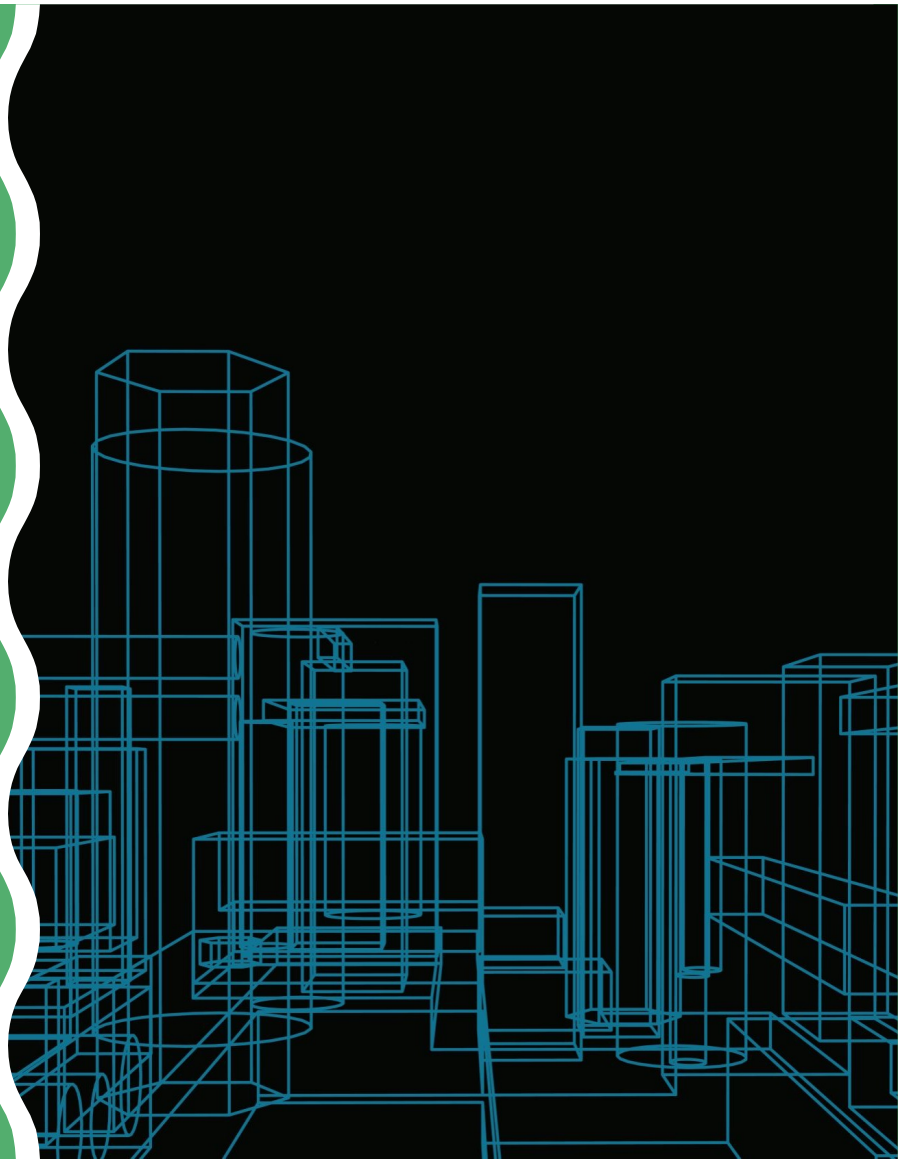
KOTLIN

LUCAS, JACK, NOLAN



WHAT IS KOTLIN?

- Kotlin is a statically typed programming language that runs on the Java Virtual Machine and can be compiled to JavaScript source code or use the LLVM compiler infrastructure.
- Developed By: JetBrains
- First Appearance: July 2011
- Official Website: https://kotlinlang.org

OVERVIEW AND HISTORY

- Origins: Developed by JetBrains, the company behind tools like IntelliJ IDEA and ReSharper.
- Purpose: To provide a more concise, expressive, and safer alternative to Java.
- Major Milestones:
  - 2010: Kotlin was developed by JetBrains
  - 2011: 'Project Kotlin' was released
  - 2016: Kotlin v1.0 released
  - 2017: Google announces first-class support for Kotlin on Android
  - 2019: Kotlin v1.3 released with coroutines for asynchronous programming

DESIGN GOALS

- **Concise**: Reduced boilerplate code compared to Java.
  - Use of fewer lines and less repetition
- **Safe**: Avoid entire classes of errors such as null pointer exceptions.
  - Less prone to runtime errors
- **Interoperable**: Full operability with Java code, allowing use of all existing Java libraries.
  - Both target the Java Virtual Machine (JVM)
- **Tool-friendly**: Designed to be a language that works well with modern development tools.
  - Give developers a smooth and productive experience when working with the language

LANGUAGE FEATURES

- **Static Typing**: Kotlin is statically typed which helps catch errors at compile-time.
- **Null Safety**: Variables are non-null by default; nullability is explicitly declared.
- **Extension Functions**: Ability to extend a class with new functionality without inheriting from the class.
- **Smart Casts**: Automatic casting of types if they are logically checkable.
- **Data Classes**: Simplifying the creation of classes that just hold data.
- **Coroutines**: Efficient and comprehensive framework for asynchronous programming.

BINDING, DATA TYPES, CONTROL STATEMENTS, AND MEMORY ALLOCATION

- **Binding Choices**: Supports both static and dynamic types while emphasizing static types with inference for conciseness and safety.
- **Data Types**: Includes integers, floating-point numbers, characters, booleans, arrays, and strings with more nuanced control than Java.
- **Control Statements**: Similar to Java but includes enhancements like 'when' for more expressive conditional statements.
- **Memory Management**: Automatic memory management via garbage collection, similar to Java.

OBJECT ORIENTED LANGUAGE

- Allows classes, inheritance, interfaces and more
  - Define a class using the class keyword
- Allows the use of Data Structures
- Every class has a super class called Any
  - Any is the root of the Kotlin class
  - Contains methods equals(), hashCode(), and toString()

WHY USE KOTLIN?

- Java → Kotlin: An easy transition from Java to Kotlin, compiles to JVM and uses existing Java libraries and frameworks
- Kotlin is normally associated with Android development, but it's a multi-platform language that can be used for web, server-side, and desktop application development
- Supported by several IDEs including IntelliJ IDEA, Android Studio, and Eclipse
- Officially supported language for Android development by Google

KOTLIN APPLICATION USES

- Cross-platform Mobile Applications
  - Kotlin Multiplatform: Allows sharing of code between iOS and Android.
  - Example Companies: VMware, Netflix
- Android Application Development
  - Preferred Language: Officially supported and recommended by Google.
  - Example Companies: Google, Trello, Evernote, Coursera
- Web Application Development
  - Framework: Developed by JetBrains for building asynchronous servers and clients.
  - Example Companies: JetBrains, PostMates
- Desktop Application Development
  - Framework: Kotlin framework for developing desktop applications on JavaFX.
  - Example Companies: Amazon (internal tools), Intuit (financial software)
- Data Science Applications
  - Kotlin for Data Science: Increasingly used due to its interoperability with Java and simplicity.
  - Example Companies: JetBrains (internal data analysis), Agoda (data processing)

KOTLIN INTEROPERABILITY

- Can call from Kotlin into Java code and vice versa
- Both Kotlin and Java target the Java Virtual Machine (JVM)
- Projects in Kotlin and Java may work together fluently
- Kotlin can make use of Java tools, libraries, and frameworks
- A main reason for Kotlin's success

CHALLENGES WITH KOTLIN

- Compilation Speed
  - due to the additional overhead of Kotlin's features, such as null safety checks and extension function
- Adoption Outside Android
  - Kotlin adoption in other areas, such as desktop and enterprise applications, is still catching up
- Job Market
  - The job market for Kotlin developers, especially outside of Android, is still smaller compared to established languages like Java, C#, and JavaScript

CLASSES

JAVA

```java
public class Animal {

    private String eats;
    private int noOfLegs;

    public Animal( String food, int legs){
        this.eats = food;
        this.noOfLegs = legs;
    }

    public String getEats() {
        return eats;
    }

    public void setEats(String eats) {
        this.eats = eats;
    }

    public int getNoOfLegs() {
        return noOfLegs;
    }

    public void setNoOfLegs(int noOfLegs) {
        this.noOfLegs = noOfLegs;
    }

}
```

KOTLIN

```kotlin
class Animal(private var eats: String, private var noOfLegs: Int) {

  fun getEats(): String {
    return eats
  }

  fun setEats(eats: String) {
    this.eats = eats
  }

  fun getNoOfLegs(): Int {
    return noOfLegs
  }

fun setNoOfLegs(noOfLegs: Int) {
    this.noOfLegs = noOfLegs
  }
}
```

BOTH CODES CREATE THE SAME ANIMAL CLASS

CODE : JAVA VS KOTLIN

JAVA

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

KOTLIN

```kotlin
fun main() {
    println("Hello World!")
}
```

BOTH CODES PRINT HELLO WORLD!

INHERITANCE

JAVA

```java
public class Cat extends Animal {
    private String color;

    public Cat(String eats, int noOfLegs, String color) {
        super(eats, noOfLegs);
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

KOTLIN

```kotlin
class Cat(veg: Boolean, food: String, legs: Int, private var color: String) : Animal(food, legs) {

  fun getColor(): String {
    return color
  }

  fun setColor(color: String) {
    this.color = color
  }
}
```

CREATES CAT CLASS BY INHERITING ANIMAL CLASS

DEMO

SOURCES

- https://kotlinlang.org/docs/home.html
- https://blog.jetbrains.com/kotlin/2014/12/javascript-interop/
- https://kotlinlang.org/docs/extensions.html
- https://en.wikipedia.org/wiki/Coroutine