Android Bootcamp with Kotlin

WELCOME!

Develop Intelligence

Nate Ebel

Android Developer & Instructor

Join Us in Making Learning Technology Easier

Develop Intelligence

Our mission...

Over 16 years ago, we embarked on a journey to improve the world by making learning technology easy and accessible to everyone.

...impacts everyone daily.

And it's working. Today, we're known for delivering customized tech learning programs that drive innovation and transform organizations.

In fact, when you talk on the phone, watch a movie, connect with friends on social media, drive a car, fly on a plane, shop online, and order a latte with your mobile app, you are experiencing the impact of our solutions.

Upskilling and Reskilling Offerings

Develop Intelligence

Intimately customized learning experiences just for your teams.

- **Workshop** — 2-3 day upskilling experiences
- **Fast Track** — 5-day reskilling experiences
- **Learning Spike** — 1-day technology overviews
- **Target Topics** — 90-minute instructor-led micro-learnings
- **Hack-a-thon** — Learn and build an MVP in 2-3 days

BACK END DEVELOPMENT

BIG DATA

CLOUD COMPUTING

DEVOPS

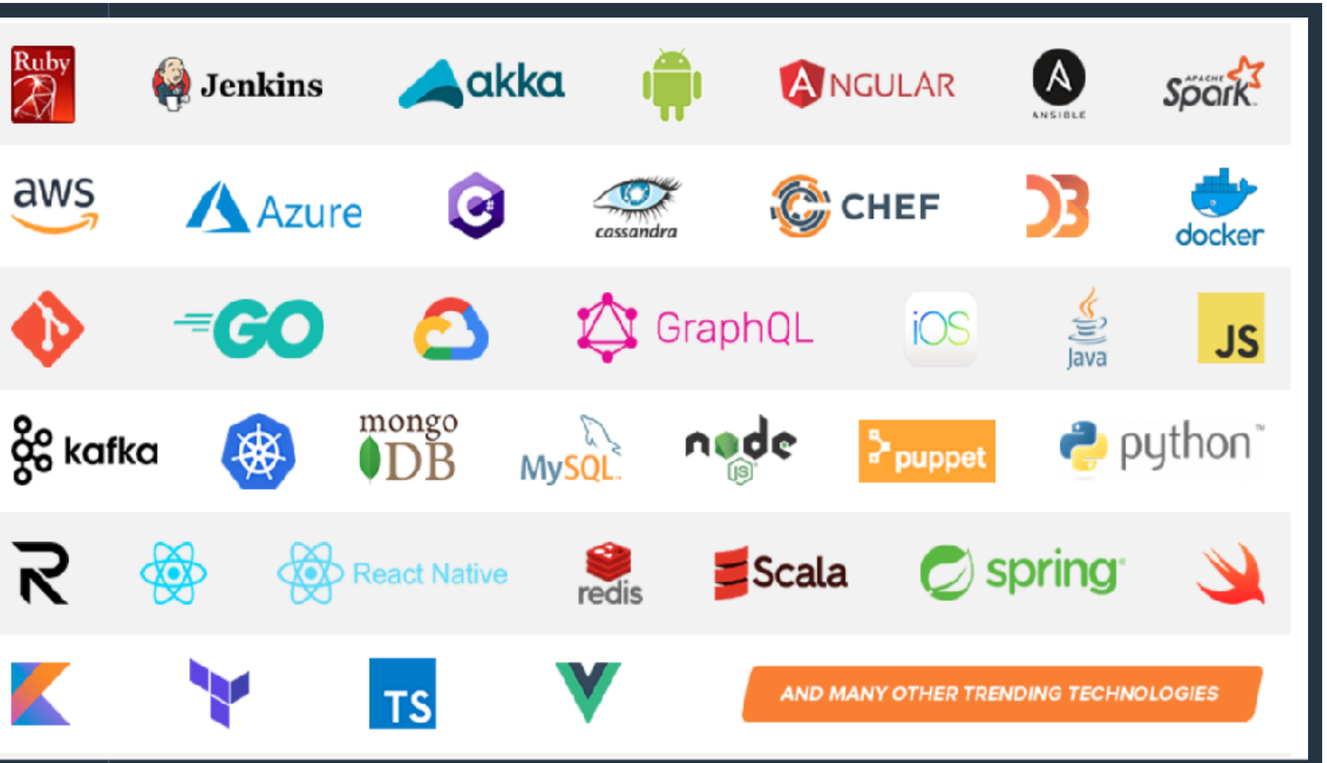
FRONT END DEVELOPMENT

MACHINE LEARNING

MOBILE APP DEVELOPMENT

SOFTWARE ENGINEERING

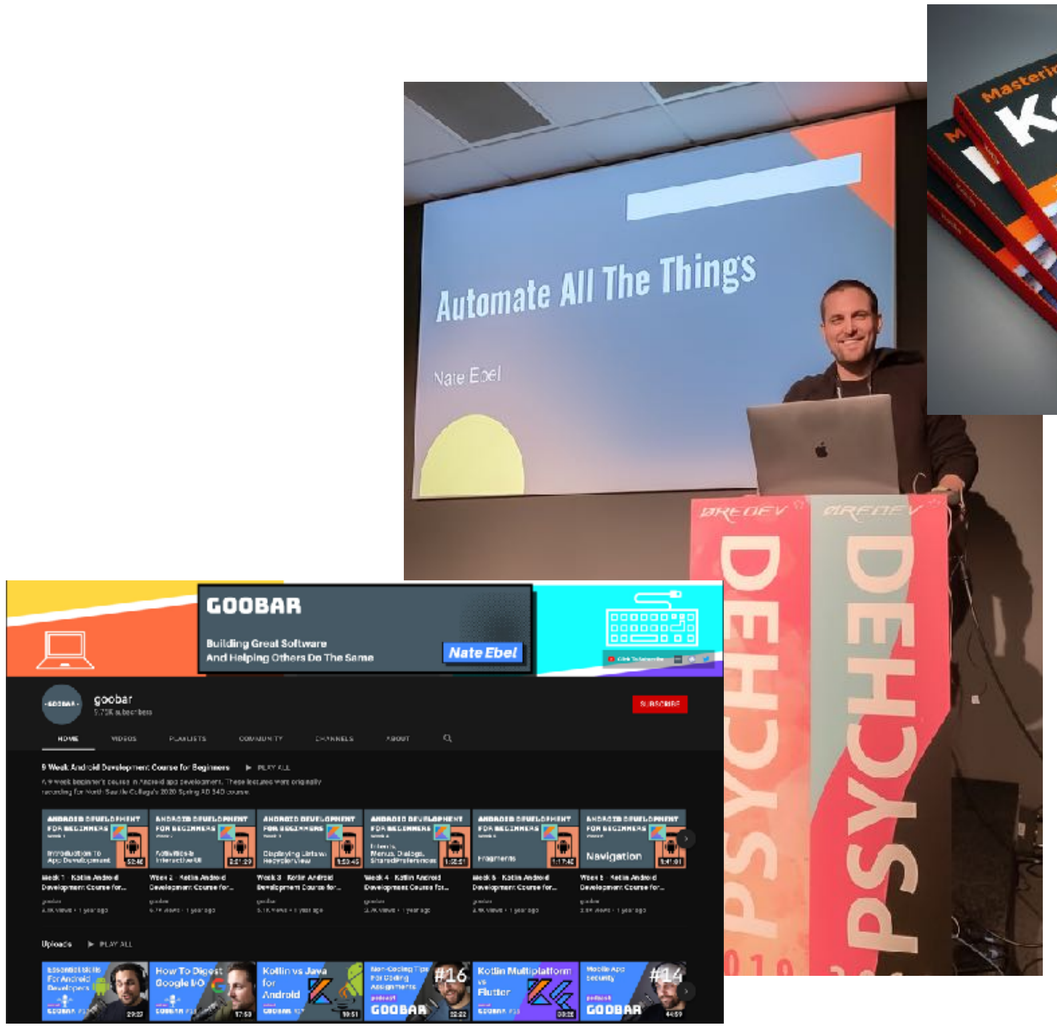
SYSTEM ADMINISTRATION



World Class Practitioners

About the Trainer

- Mobile developer
- Speaker
- Instructor
- goobar.dev
- Youtube/podcast



Android Experience

- 10 years working with Android
- 6 years working with Kotlin
- Google Developer Expert for Kotlin
- Wrote a book - “Mastering Kotlin”
- Migrated multiple production codebases to Kotlin
- Tutorials and courses on YouTube

Introductions

Introductions

- Who am I?
- What is my experience with Android?
- What is my experience with Kotlin?
- Why am I taking this training?

What to Expect this Week?

Objectives

Develop Intelligence

At the end of this course you will be able to:

- Better understand the Android ecosystem
- Write Android applications using Kotlin
- Build multi-screen, interactive user interfaces
- Understand Android app architecture, including Fragments, WorkManagers, Navigation and other Android components
- Setup unit tests, debug apps, and integrate external dependencies in Android Studio

What's the Transformation?

By the end of this week:

- You will be an Android developer
- You will have built multiple applications
- You will have experience with a modern Android development toolset
- You will understand how to build, deploy, and debug Android apps
- You will understand the challenges and opportunities involved with modern Android development

Why Android development with Kotlin?

- Mobile apps are ubiquitous
- Android is the most widely used mobile operating system in the world
  - https://gs.statcounter.com/os-market-share/mobile/worldwide
- Kotlin is the preferred language for Android since 2019

High-level Topics

Develop Intelligence

- What is the state of Android development today?
- How to setup your Android development environment?
- How to create a new Android app?
- How to write Kotlin code?
- How to build user interfaces?
- How to navigate through an application?

High-level Topics

Develop Intelligence

- What is MVVM architecture within the context of Android?
- How to perform long-running background work?
- How to persist local data?
- How to fetch and deserialize network data?
- How to mange application permissions?
- How to write unit tests?

Schedule for the week

Develop Intelligence

- Monday - Foundations
- Tuesday - Building Interactive Applications
- Wednesday - Application Architecture
- Thursday - Performing Background Work
- Friday - Special Topics

Roadmap for the day

9-9:45	Lesson 1
9:45-10	Break
10-10:45	Lesson 2
10:45-11	Break
11-12pm	Lesson 3
12pm-1pm	Lunch
1-1:45	Lesson 4
1:45-2	Break
2-2:45	Lesson 5
2:45-3	Break
3-4	Lesson 6
4-5	Office Hours

- This is a general outline for the day
- We will adjust as needed

What to expect during the day?

- We'll include an interactive mix of lecture and labs
- Most sections will include a lab
- The expectation is that we will work through the lab together as we move through the section
- With 1-2 "challenges" to complete on your own at the end of the section
- We will reserve time at the beginning and end of each day to finish up labs and ask questions

Prerequisites

- Familiarity with any object-oriented programming language, such as Java, JavaScript, or C#
- A basic understanding of git and GitHub

Technical Requirements

- JDK 11 or JDK 14
- Android Studio 4.2+
- Access to GitHub

Logistics

Note About Virtual Trainings

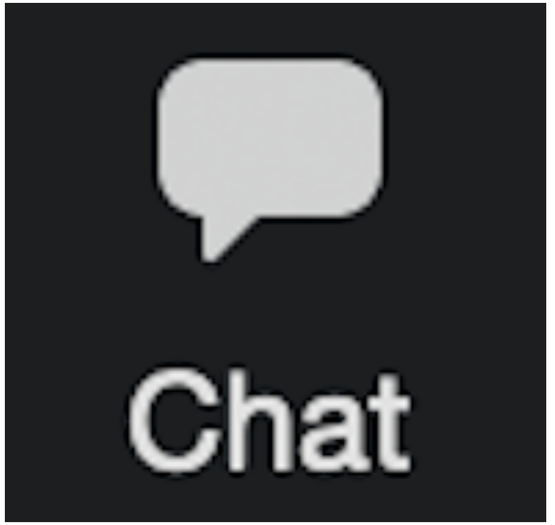
What we want

...what we've got

Virtual Training Expectations for You

Arrive on time / return on time

Mute unless speaking



Use chat or ask questions verbally

Virtual Training Expectations for Me

I pledge to:

- Make this as interesting and interactive as possible
- Ask questions in order to stimulate discussion
- Use whatever resources I have at hand to explain the material
- Try my best to manage verbal responses so that everyone who wants to speak can do so
- Use an on-screen timer for breaks so you know when to be back

What if I have a problem?

- Send a message in the Slack channel
- Contact Tanya at DI
- Reach out to me via Slack or online (@n8ebel)

Where to find the materials?

Develop Intelligence

- Code and Labs found on GitHub
- https://github.com/goobar-dev

Post-training resources

- What will be available after the training?
  - Code and Labs will remain available for 1 weeks
  - Recommend forking, or otherwise copying, the repo
- What if you want a recording?
  - Contact Tanya at DI if you would like a copy of the video recording

Any Questions?

Android with Kotlin Bootcamp

Day 1 - Android Foundations

Nate Ebel

Android Developer & Instructor

@n8ebel goobar.dev

Introduction to Android

Nate Ebel

Android Developer & Instructor

@n8ebel goobar.dev

Overview

State of Android today

Challenges and opportunities of Android

Android ecosystem

Android development tools

Distributing Android applications

Mobile Is Everywhere

Mobile Operating Systems

iOS

Android

A Brief History of Android

2003 - Android Inc building camera OS

2005 - Acquired by Google

2007 - v1.0 Beta

2008 - G1 Released

2009 - Cupcake released

2014 - Android 5.0 Lollipop introduces Material Design

2020 - Android 11

Android Open Source Project

Core of all Android variants

Build on a Linux Kernel

Directed by Google

Open for contribution

https://source.android.com/

Android vs AOSP

Android Variants

AOSP

Google’s Android

Fire OS

OxygenOS

HarmonyOS

Android OS Architecture

Apps

SDK

Native C/C++ Libs & Android Runtime

Hardware Abstraction Layer

Linux Kernel

https://developer.android.com/guide/platform

Android SDK

APIs

Tooling

Emulators

Documentation

SDK + Libraries

How Can We Build Apps?

Native Apps

Cross Platform

Thin Web Client

PWA

Mobile Web

Mobile vs Web

Mobile Native

Direct access to sensors & apis

Highly performant

Offline experiences

Slower release cycles

Mobile Web

Browser are even more ubiquitous

More abstraction

Single codebase

Faster release cycles

What Does Native Android Development Look Like Today?

Android Studio / IntelliJ

Kotlin/Java/C++

Gradle

Where Do Apps Run?

Phone / Tablet

Kindle

ChromeOS

Windows 11

Watches/Auto/IoT

How can we distribute our apps?

APK or Bundle

Google Play

Other marketplaces

Side Loading

Challenges of Android Development

Manufacturers

OS Variants

Form Factors

API Versions

Languages

Hardware

Fragmentation is a major challenge

https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf

How to choose supported users?

https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide

Android Is a Massive Opportunity

The Android Ecosystem

Android Jetpack

Google Play Services

Material Design

3rd Party Libraries

Community Support

Setting Up Your Android Development Environment

Nate Ebel

Android Developer & Instructor

@n8ebel goobar.dev

Overview

Installing Android Studio

Creating your first project

Creating an emulator

Building and deploying your first app

How Will You Build Your Android Apps

Manually

Terminal

IDE

IDEs for Android Development

IntelliJ

Android Studio

Android Studio

Based on IntelliJ

Supports Kotlin, Java, C++

Tight integration with Android tooling

Tight integration with emulators

Leverages Gradle

OS Requirements for Android Studio

MacOS

Linux

Windows

ChromeOS

JDK Requirements

- Must have a JDK installed
- JDK supports the JVM-based build tooling
- JDK 11+ recommended as of Android Studio 4.2
- Ensure that JAVA_HOME is set on system

How to Install Android Studio?

Download Site

Jetbrains Toolbox

Installing Android SDK

Install via Android Studio

Can install multiple versions

Typically install the latest

Add *tools* and *platform-tools* to your path

Let’s Create a New Android Studio Project

Where to Run Your App?

Physical Device

Emulator

Physical Devices

Usually faster to deploy/test

Requires enabling Developer Options

More exciting

Stockpile for testing

Emulators

Install tooling via SDK Manager

Create/customize to suit your needs

Helps with testing different configurations

Let’s Deploy Our App

Gradle

What is Gradle?

Why Gradle for Android?

How to interact with Gradle?

How to build/deploy via Gradle?

How to improve Gradle performance?

Basic Build Configuration

applicationId

minSdk

targetSdk

compileSdk

Debugging Basics

Add a log statement

Set a breakpoint

Deploy debugger

Step into/over lines

Evaluate expressions


Lunch

Introduction to Kotlin

Nate Ebel

Android Developer & Instructor

@n8ebel goobar.dev

Overview

A brief history of Kotlin

Why Kotlin for Android?

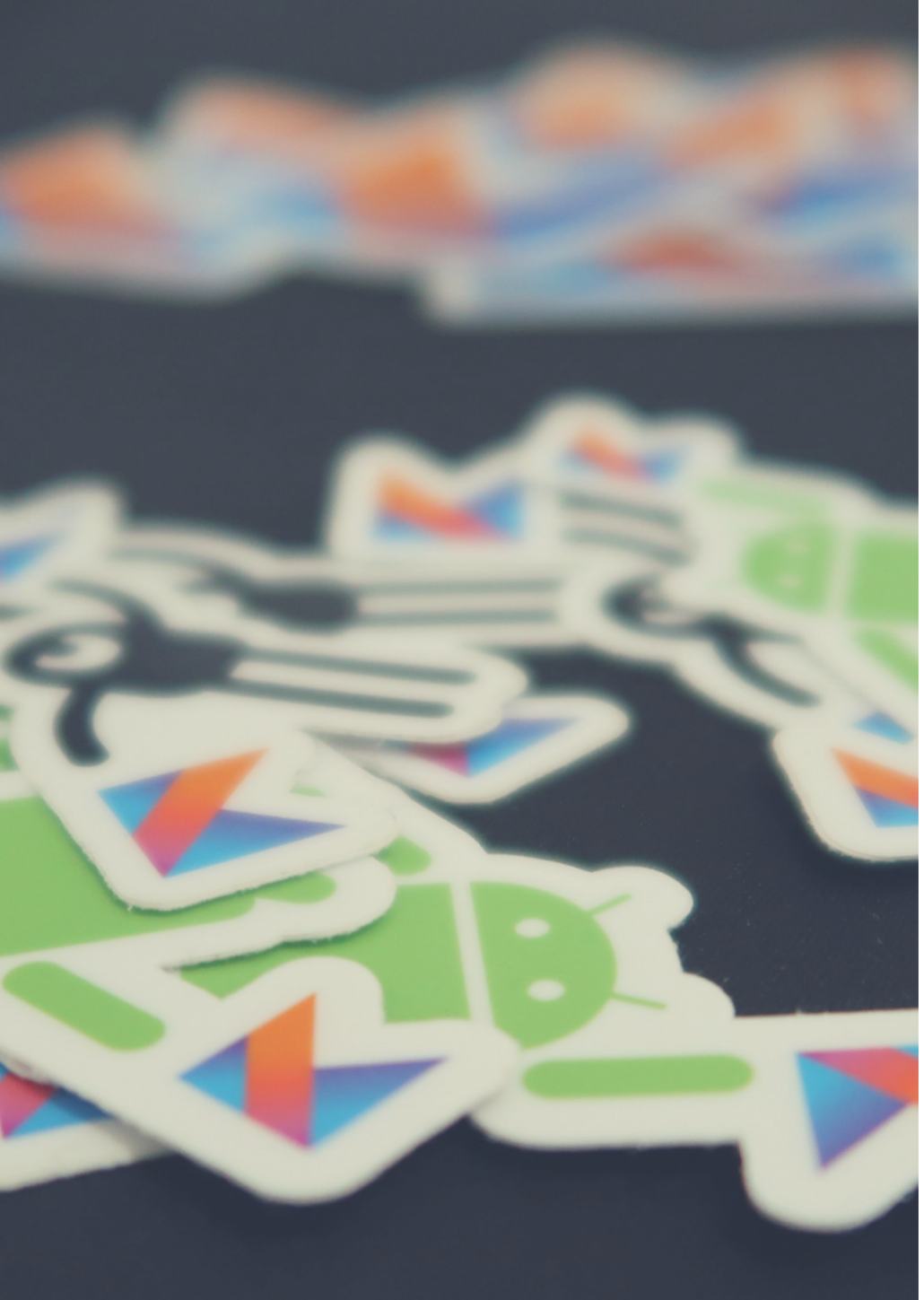
State of the Kotlin ecosystem

How to start with Kotlin?

How can Kotlin be used for Android?

Hello World in Kotlin

```
fun helloWorld() = println(“Hello World!”)
```



Kotlin started in 2010 by Jetbrains

Started gaining traction within Android in 2015

1.0 release in 2016

Google started supporting in 2017

Google goes “Kotlin First” in 2019

Kotlin 1.5 is latest stable

Why Kotlin?

JVM

Modern Features

Standard Library

Flexible

Why Kotlin for Android?

Kotlin is 100% interoperable with Java, brings more modern languages features than older Java versions, and is the Google-recommended language for Android.

State of the Kotlin Ecosystem

1 major release a year

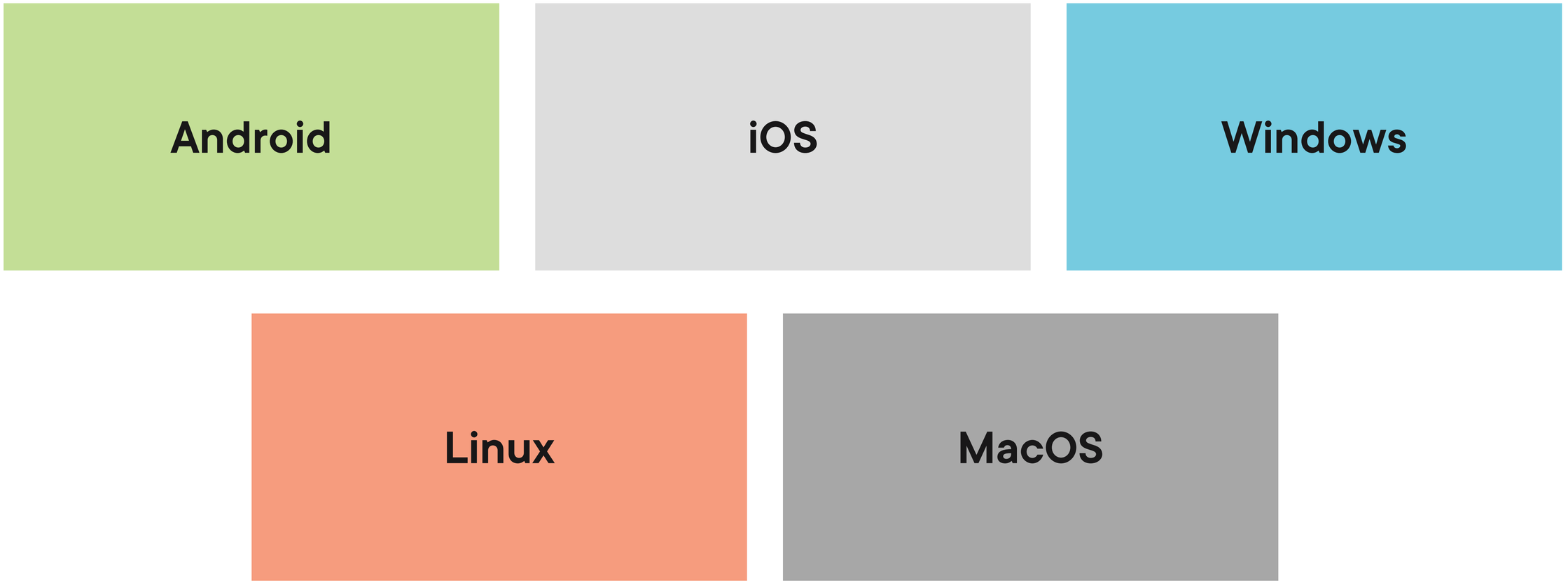
Multiple small releases

1.5.20 is latest version

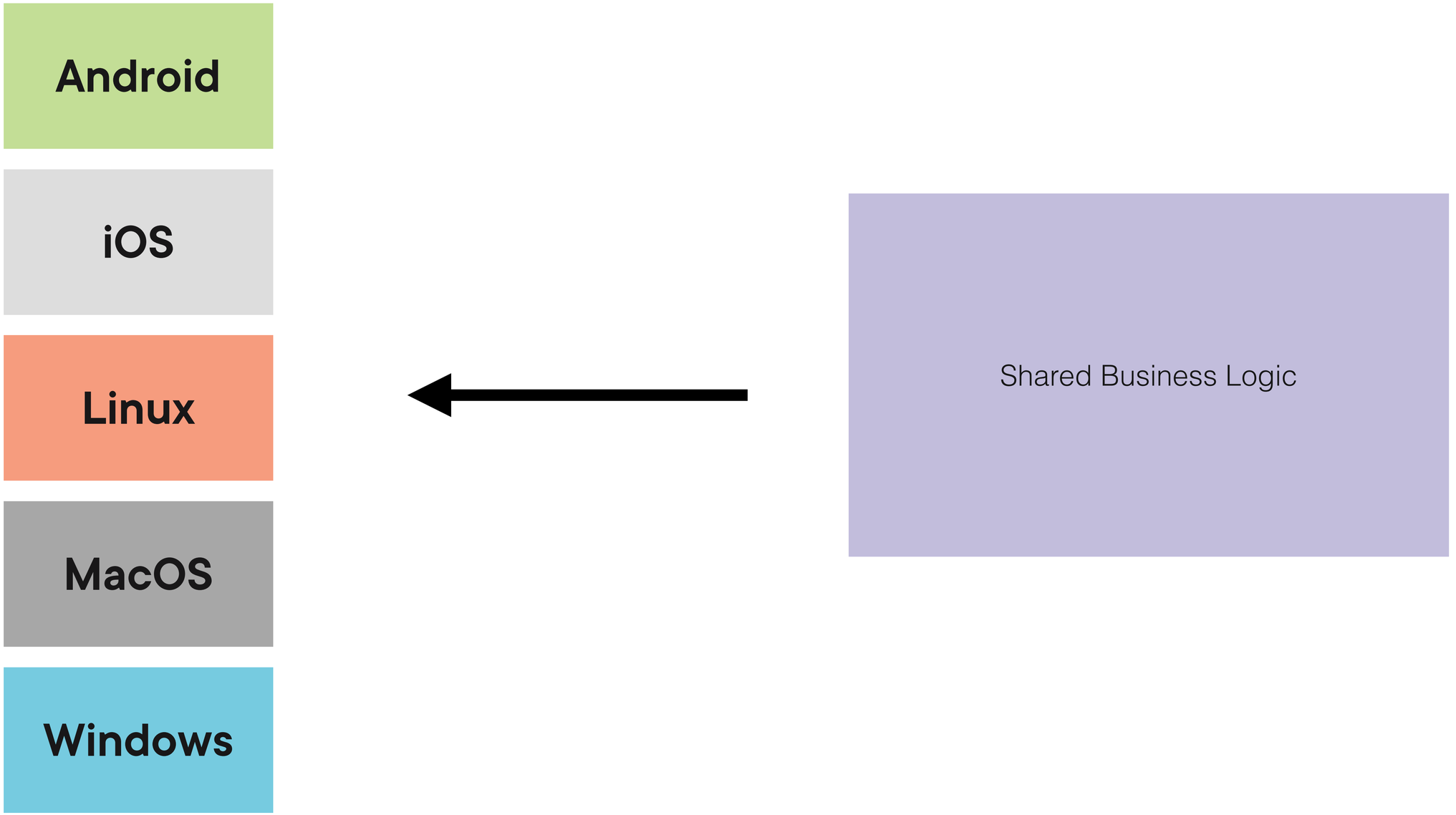
Fully compatible with JVM ecosystem

Used for Android, backend, and Multiplatform

Kotlin Multiplatform



Kotlin Multiplatform



How to Get Started with Kotlin?

Koans

Scratch File

Android Studio

```
fun helloWorld() {
    println(“Hello World!”)
}


fun main() {
    helloWorld()
}

output:
“Hello World!"
```

- ***fun*** **keyword denotes a function**
- ***println()*** **prints to System.out**

- ***main()*** **is entrypoint for pure Kotlin app**
- **invoke a function**

```
fun helloWorld(greeting: String) {
    println(“$greeting World!”)
}


fun main() {
    helloWorld(“Hey”)
}

output:
“Hey World!"
```

- Add a *greeting* parameter of type String
- String template substitution

- Pass argument to the function

Where to Use Kotlin for Android?

Application code

Multiplatform code

Buildscript logic

Buildscript plugins

Server-side code

Kotlin Fundamentals

Nate Ebel

Android Developer & Instructor

@n8ebel goobar.dev

Overview

First-class functions

Interfaces

Classes

Companion Objects

Collections

Common Data Types in Kotlin

String

Int

Boolean

Float

List

Unit

More Data Types

Byte

Uint, Long, Short

Char

Double

Array, IntArray, LongArray, ShortArray

Nothing

Pair, Tripple

Throwable

Non-Nullable Types

Kotlin differentiates between null & non-null

Must explicitly mark something as nullable

```
var name: String = “Nate”
name = null  <– ERROR
```

- Create a non-nullable variable
- Cannot assign a null value to non-null type

```
var name: String? = “Nate”
name = null  <– OK
```

- Create nullable variable
- Ok to assign null value

Functional Types

Functions are first-class citizens

Can assign function to a variable

Can pass functions as function parameters

Functions can exist outside of enclosing classes

Intermediate Functions

Default parameter values

Named arguments

***vararg* parameters**

Advanced Functions

Higher-order functions

Trailing lambda syntax

Extension functions

Common Collection Types

List

Map

Set

MutableList

MutableMap

MutableSet

Operating On Collections

forEach{}

map{}

filter{}

first{}

find{}

Object Oriented Programming in Kotlin

Classes

Interfaces

Enum Classes

Data Classes

Sealed Classes

Object Classes

Connecting to Android Views

Nate Ebel

Android Developer & Instructor

@n8ebel goobar.dev

Overview

Android View system

TextViews and Buttons

Working with Views in code

Responding to user actions

Displaying simple feedback to users

How Do We Build a UI in Android?

Building an Android UI

Layouts represent all/part of what is on screen

Views & ViewGroups are added to a layout

Layouts are loaded, parsed, and inflated into concrete classes

We can reference and interact with these classes

Layouts

How do we define a layout?

How do we preview our layout?

How do we consume our layout?

Basic Views

Basic ViewGroups

FrameLayout

LinearLayout

ConstraintLayout

FrameLayout

Most basic

No explicit ordering or constraints

Very efficient

LinearLayout

Draws Views one after the other

Vertical or Horizontal

Efficient

Great for stacked elements

ConstraintLayout

Most complex

Very powerful set of constraints

Highly efficient when done correctly

How Do We Programmatically Control Views?

Working With Views in Code

How do we programmatically connect to a View?

How do we set data into a View?

How do we interact with elements on screen?

How do we update View visibility?

How do we show basic feedback to the user?

💡 Office Hours