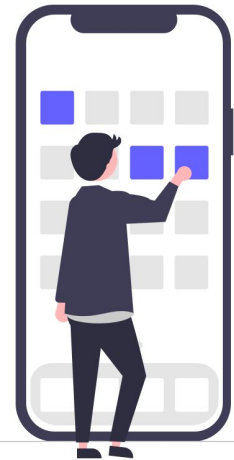


Dan Ilies  
Head of Mobile Development @Wolfpack Digital

# Native & Cross Platform Mobile App Development



# Agenda\_

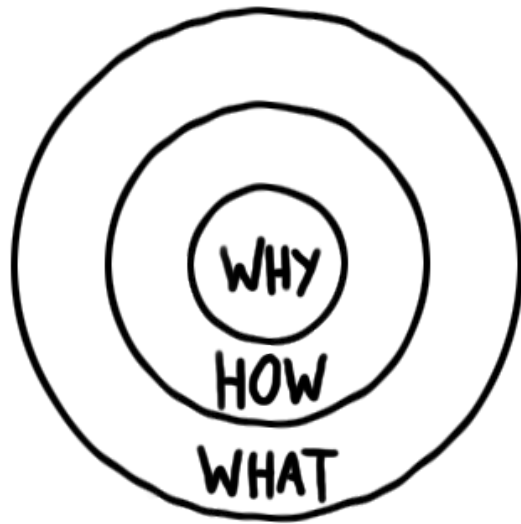
**Why** should you care about building apps?

**How** can you build an app in 2023 and beyond?

- Frameworks, programming languages
- Technical deep dive

**What** should you choose, as a:

- Developer working for a company
- Freelancer
- Business owner



# Why should you care? \_

🕒 Time spent on the phone: > **3h per day**, ~90% in apps

📱 Apps on the phone: > **60 apps**

🛍️ Shopping time: > **50%** prefer **mobile apps**

💰 Revenue: > **100 \$bn per year** since 2020



Data Source: <https://buildfire.com/app-statistics/> and  
<https://www.appventurez.com/blog/mobile-app-statistics>

# Why should you build **outstanding** apps? \_

📱 App count: **>3.5M** on Android, **>2M** on iOS

🚀 New apps: **100k per month**

📅 Apps used per day: **9 to 10**

🕒 Retention: **>80%** of apps not used after **72 hours**



**It's still a great time to  
build a mobile app!**

# How can you build an app in 2023+? \_

**01**

**Native Apps**

**02**

**Cross Platform Apps**

**03**

**Hybrid Apps**

**04**

**Progressive Web Apps (PWA)**

# Native Apps\_

Tailored to each platform's needs.

- Use technology that is specific to either iOS or Android
- Offer excellent performance and full access to a platform's capabilities
- Require separate projects and code for iOS and Android



# Cross Platform Apps\_

## iOS and Android app from the same codebase

- Either create a bridge to native code, or render components from scratch.
- Use a variety of programming languages
- Still have good performance and user experience

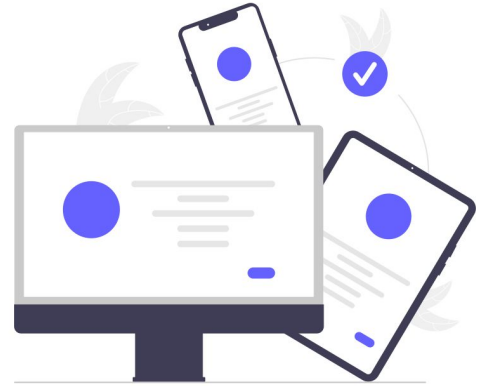




# Hybrid Apps\_

A blend between mobile and web

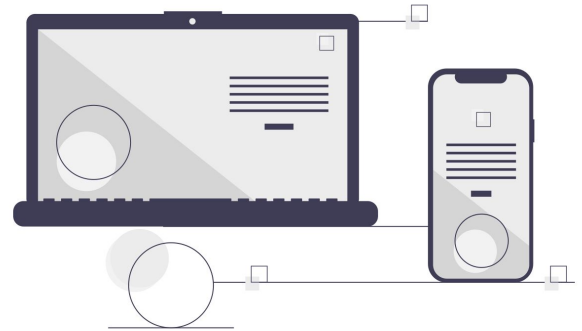
- HTML, CSS, JS, wrapped in a native app
- Use WebViews to display the UI
- Have access to native features, unlike a classic web app



# Progressive Web Apps\_

Web Apps that are adapted to mobile

- Not delivered through the classic App Store
- Similar to a website that works offline and can receive notifications
- Have all the features of a mobile browser



# Outstanding Apps

**Native Apps**

**Cross Platform  
Apps**

**Hybrid Apps**


**Progressive  
Web Apps**

# Outstanding Apps

 Performance

 User Experience

 Functionality and flexibility

 Stability and Reliability

**Native Apps**

**Cross Platform  
Apps**

**Hybrid Apps**


**Progressive  
Web Apps**

# Outstanding Apps

 Performance

 User Experience

 Functionality and flexibility

 Stability and Reliability

**Native Apps**

**Cross Platform  
Apps**

**Hybrid Apps**

**Progressive  
Web Apps**

# **Native App Development**

# Native App Development\_

## ios

**SDK** available since **2008**

Programming language: **Swift** / Objective-C

Tools: **Xcode** + a **Mac**



## Android

**SDK** available since **2008**

Programming language: **Kotlin** / Java

Tools: **Android Studio** + decent PC or Mac



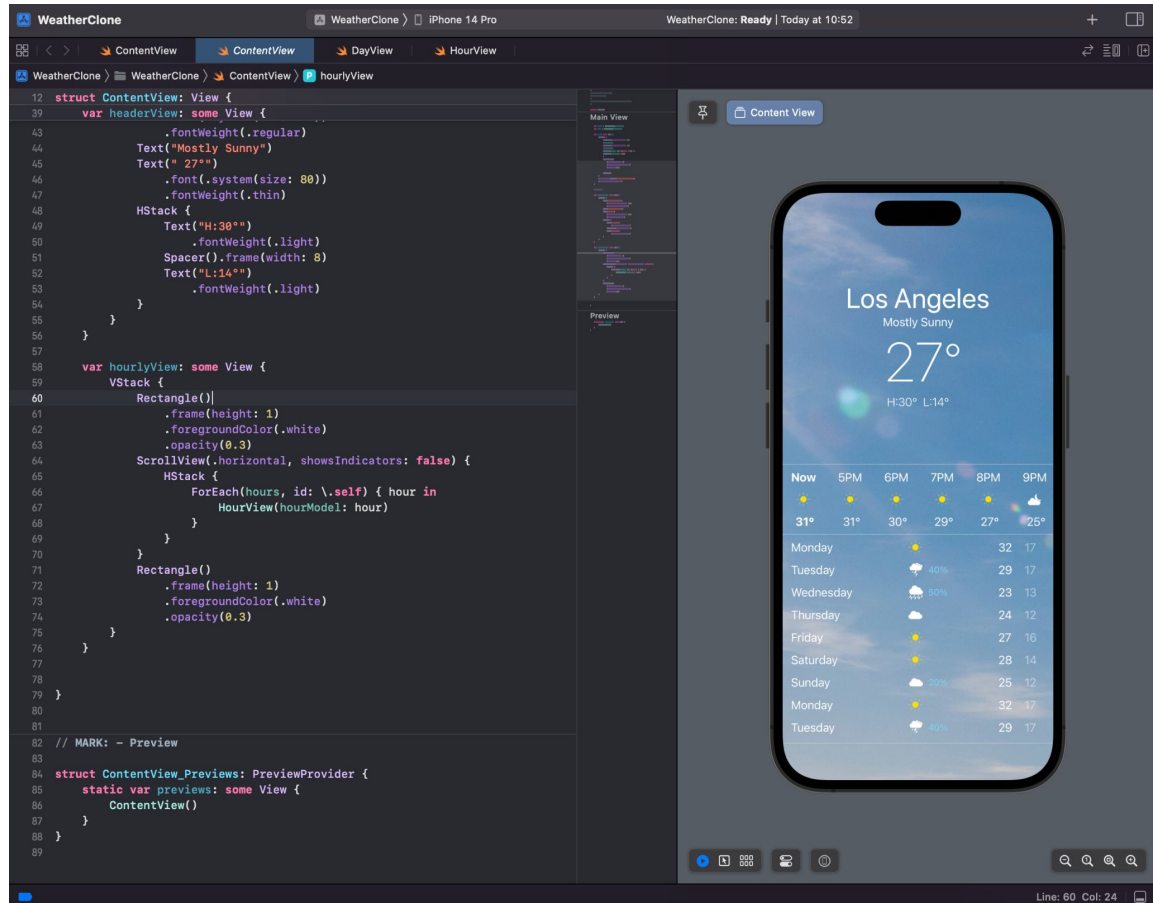
# ios\_

## Swift

- 2014
- Optionals, Value & Ref types, protocols, extensions, async/await
- Objective-C runtime
- Automatic Reference Counting

## UI Side

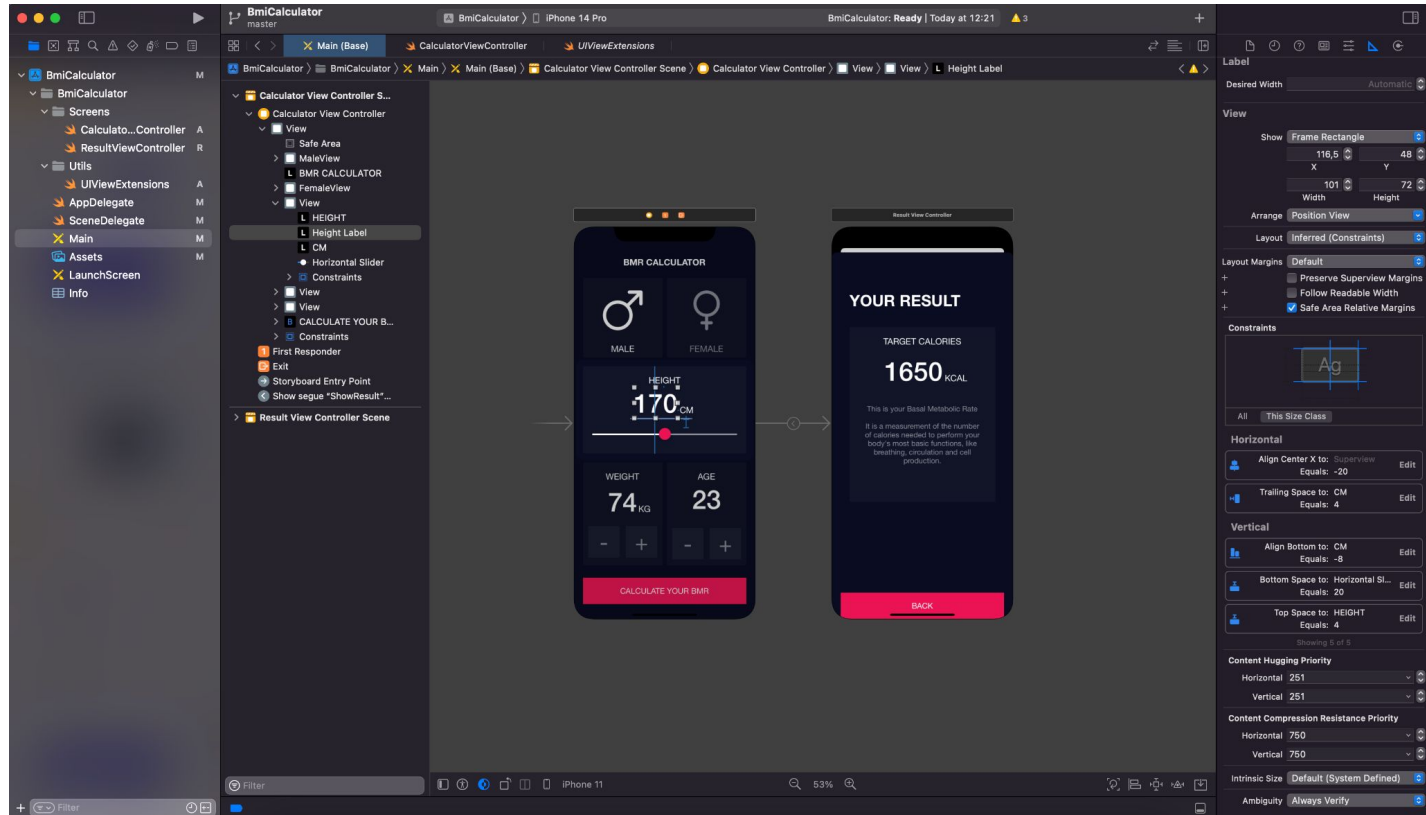
- SwiftUI (2019)
- UIKit





# Xcode

- **Debugging** & iOS specific functionality
- **Instruments** for resource inspection
- App **Signing**
- **Release** Process



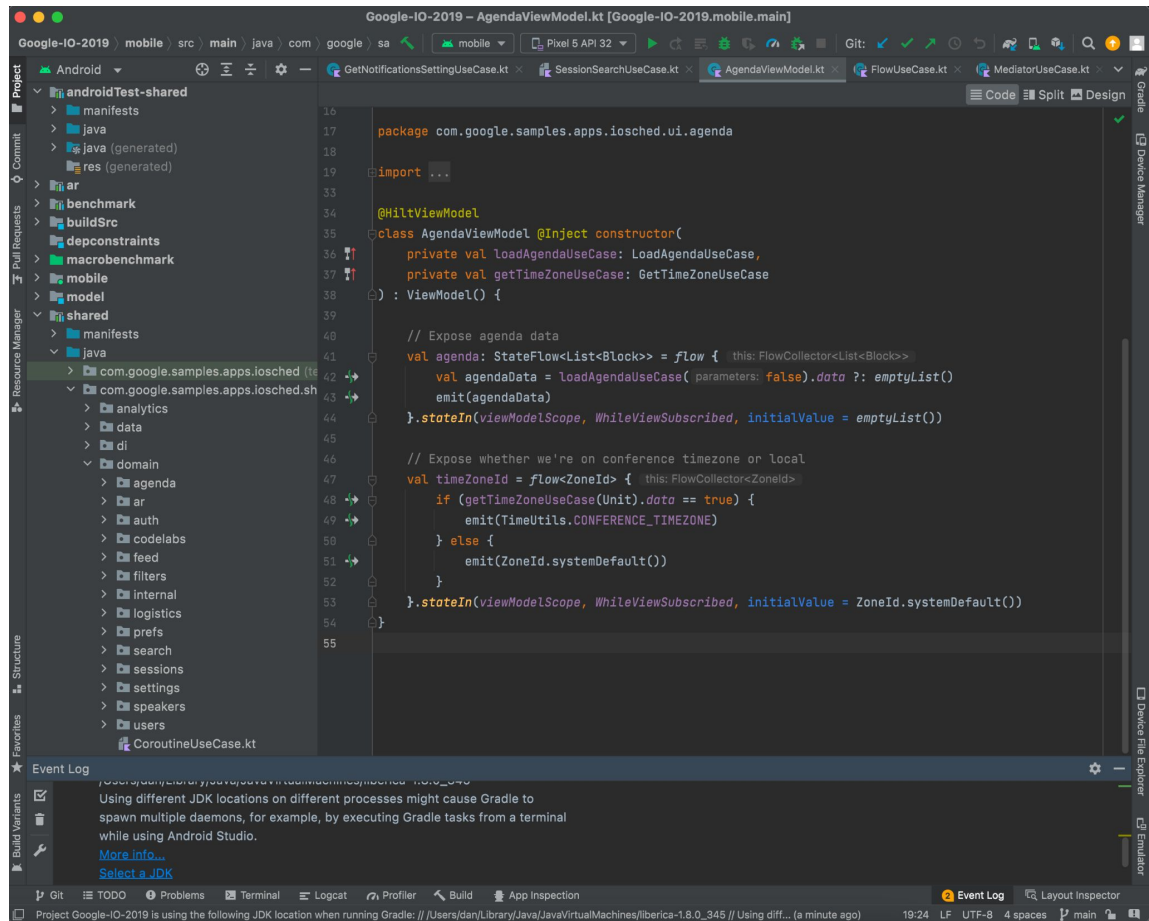
# Android

## Kotlin

- 2011 / 2019
- Optionals, compact syntax, function types, coroutines
- JVM
- Garbage collector

## UI Side

- XML Files
- Jetpack Compose (2021)



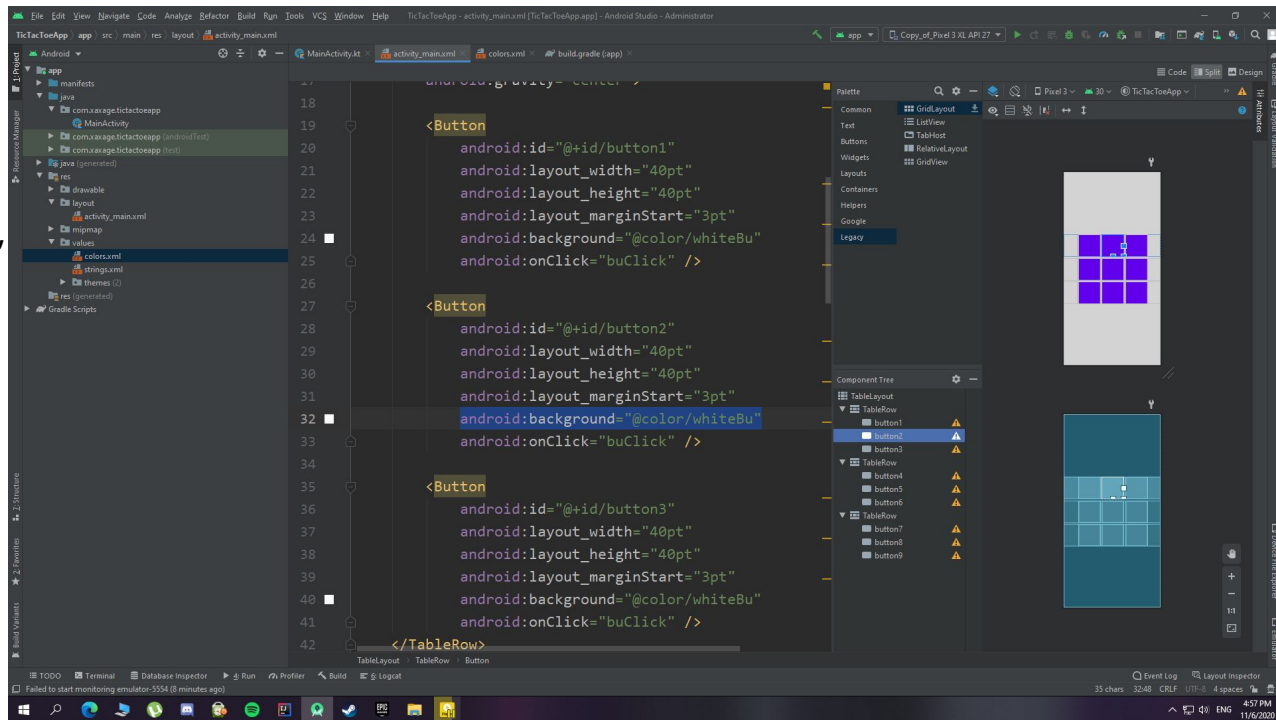
# Android\_

## Kotlin

- 2011 / 2019
- Optionals, compact syntax, function types, coroutines
- JVM
- Garbage collector

## UI Side

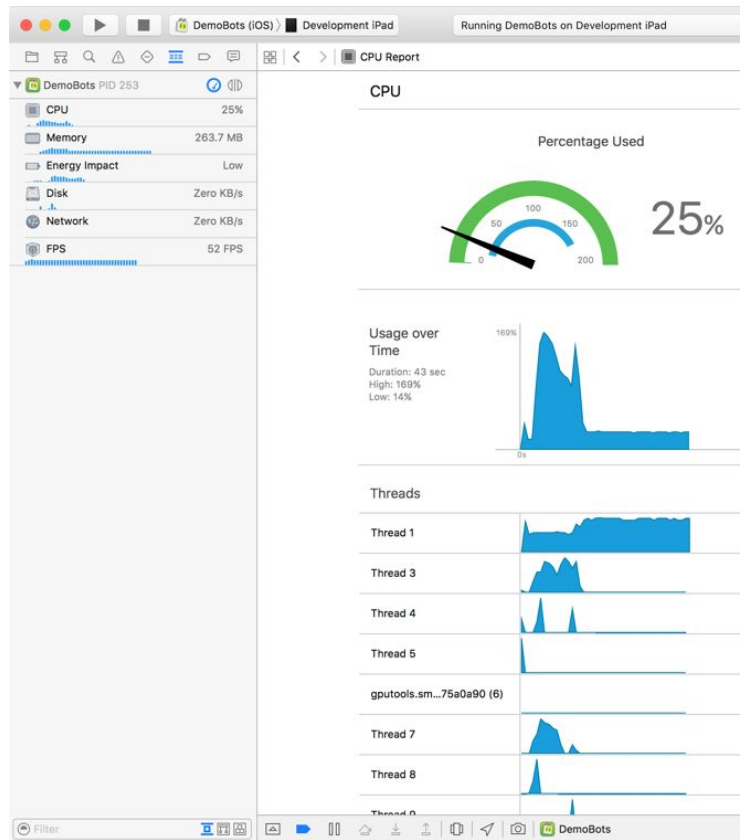
- XML Files
- Jetpack Compose (2021)



# Native Apps – Performance

As much as you can get, on a mobile device

- Take advantage of each platform's memory management strategy
- Use native components directly, with no bridges
- Closely monitor performance from the IDEs



# Native Apps – User Experience

## Consistent design with each platform's guidelines

- Use native UI components, transitions and effects
- Full control for customization if needed
- Optimization for different screen sizes and devices

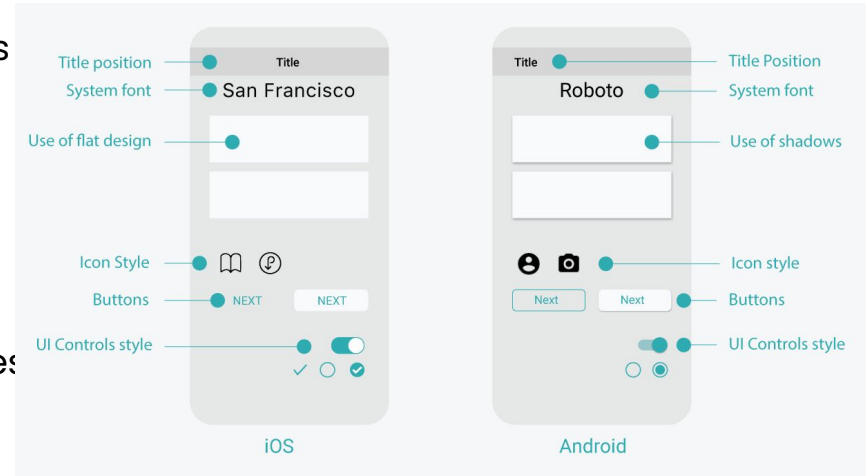


Photo source: <https://designflyover.com/>

# Native Apps – Functionality\_

## Full access to each Operating System's capabilities

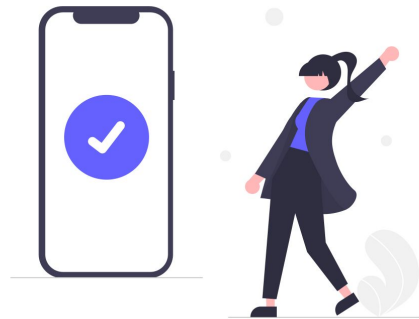
- Official SDKs for: GPS, Camera, Bluetooth, Offline Storage, etc
- Many 3rd party SDKs for integrations
- Access to platform specific features, like background modes, deep linking, etc



# Native Apps – Reliability\_

## Less potential failure points

- Immediate access to OS updates, beta versions and new features
- No bridges or other engines to also keep an eye on
- Relevant error messages and stacktraces



# Native Apps – Downsides?

It can't be all sunshine and rainbows

- No code reusability between platforms
- Skills are quite specific
- More work (and higher cost) on: project management, design, maintenance





**Cross Platform**

**App Development**

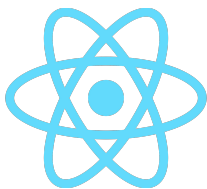
# Cross Platform Apps

## React Native

Launched in **2015** by Facebook

Programming language:  
**JavaScript**

Tools: Visual Studio Code,  
WebStorm



## Flutter

Launched in **2018** by Google

Programming language:  
**Dart**

Tools: Android Studio,  
VSCode



## Alternatives

**Xamarin** - based on **C#**,  
founded in 2011, bought by  
Microsoft in 2016

**Kotlin Multiplatform  
Mobile (KMM)** - based on  
Kotlin, alpha in 2020, beta  
in 2022, by Google



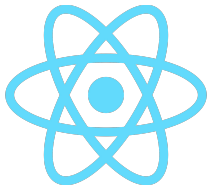
# Cross Platform Apps\_

## React Native

Launched in **2015** by Facebook

Programming language:  
**JavaScript**

Tools: Visual Studio Code,  
WebStorm



## Flutter

Launched in **2018** by Google

Programming language:  
**Dart**

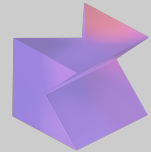
Tools: Android Studio,  
VSCode



## Alternatives

**Xamarin** - based on **C#**, founded in 2011, bought by Microsoft in 2016

**Kotlin Multiplatform Mobile (KMM)** - based on Kotlin, alpha in 2020, beta in 2022, by Google



# How does React Native work?

## Architecture

- JavaScript code running in the background on a separate thread
- Native components rendered on the UI thread

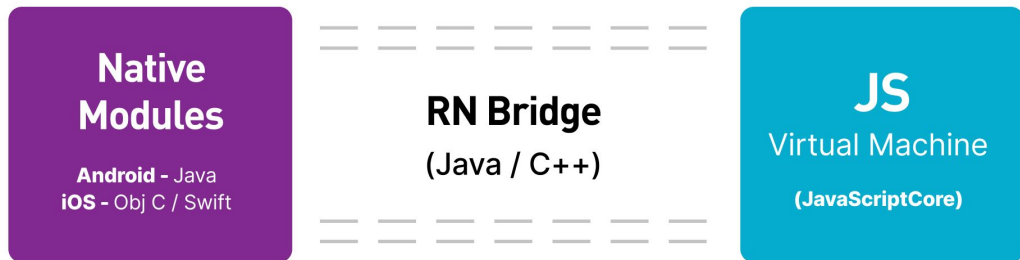


Photo Source: <https://brocoders.com>

# How does React Native work? \_

## Bridge

- A **communication layer** between the JavaScript code and the native components
- **iOS:** JavaScriptCore Engine  
**Android:** V8 Engine

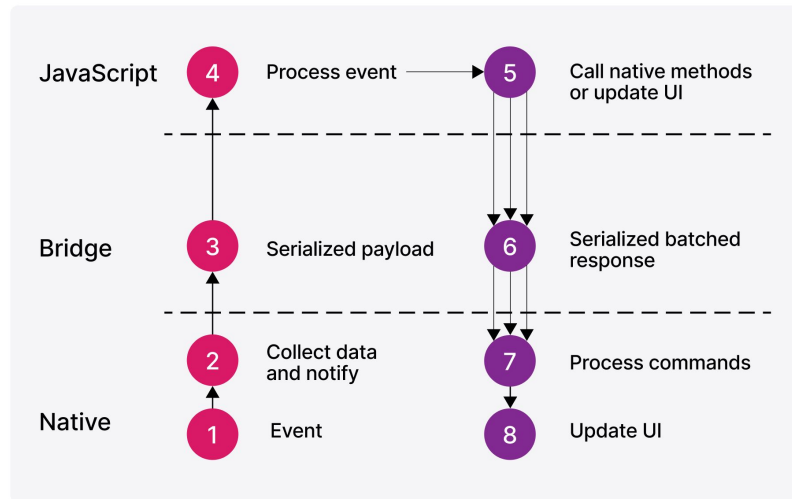


Photo Source: <https://brocoders.com>

# How does React Native work?

## Components and Rendering

→ Flexbox layout system

→ Yoga engine:  
JavaScript code → Native UI

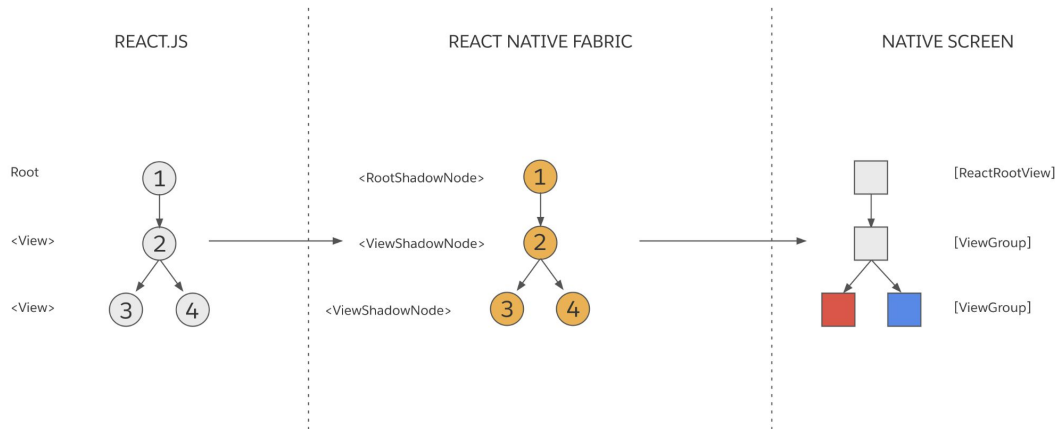
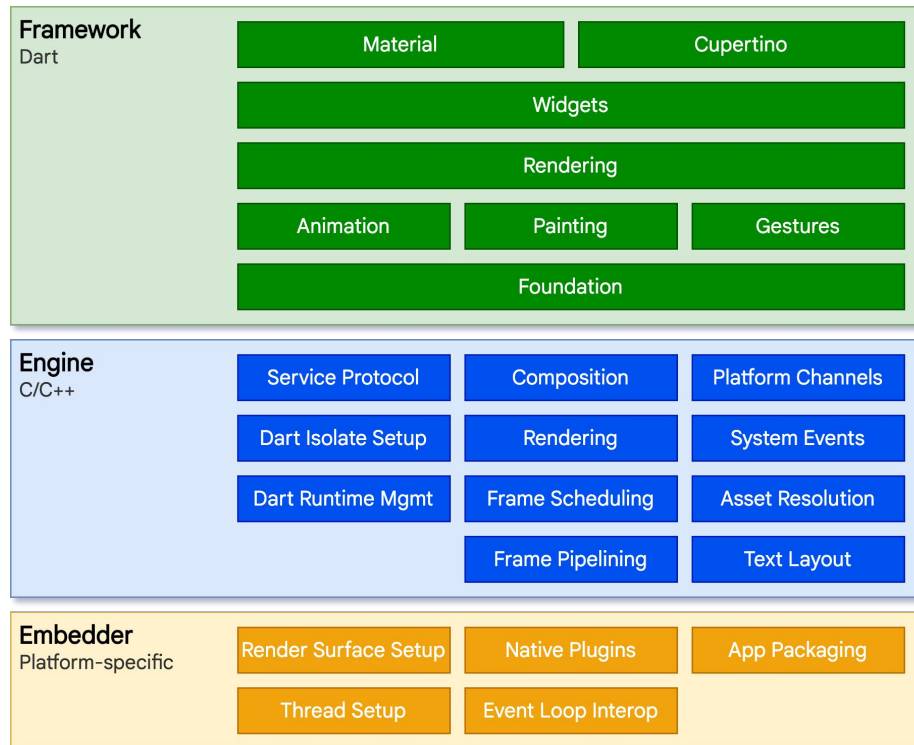


Photo Source: <https://reactnative.dev/>

# How does Flutter work? \_

## Architecture

- **Flutter engine**  
Skia Library (C/C++) → 2D rendering
- **Flutter framework**  
Dart - APIs for integrating native features



# How does Flutter work? \_

## Components and Rendering

→ **Widgets** - built-in and customizable

→ Flexible Layout System - Box Model

- Build
- Layout
- Paint

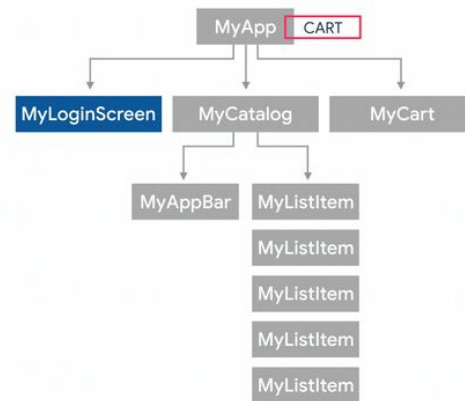
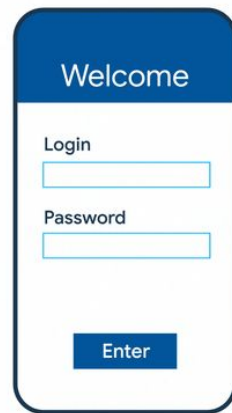
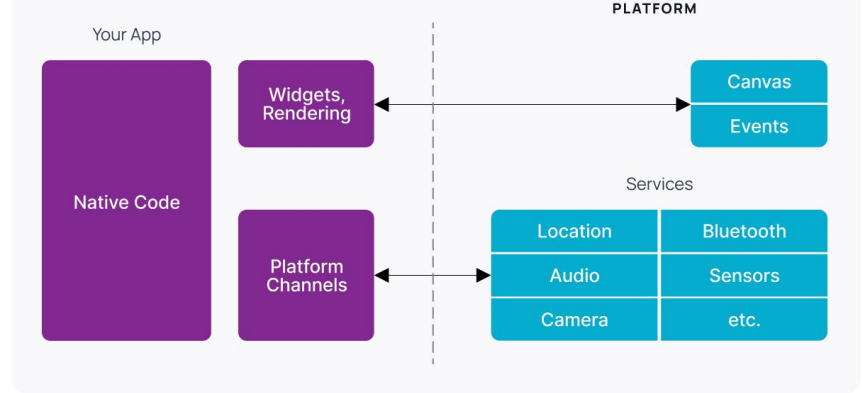
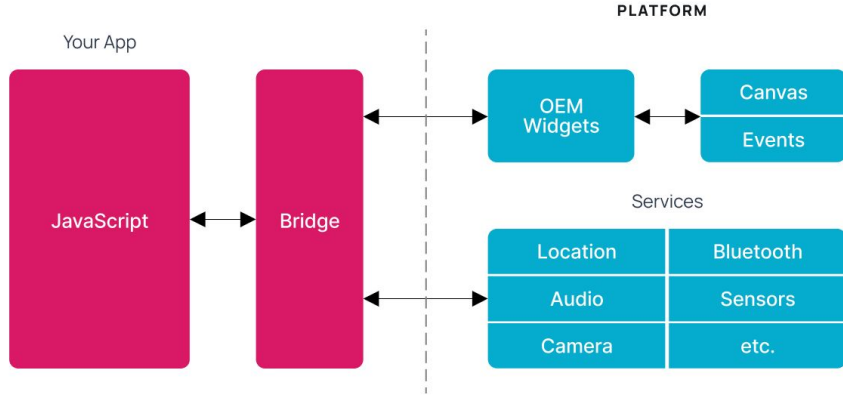


Photo Source: <https://dev.to/rubensdemelo/flutter-widget-tree-and-state-management-31an>



# Side by side\_

## React Native & Flutter



# Side by side\_

## React Native & Flutter

🚀 Performance:

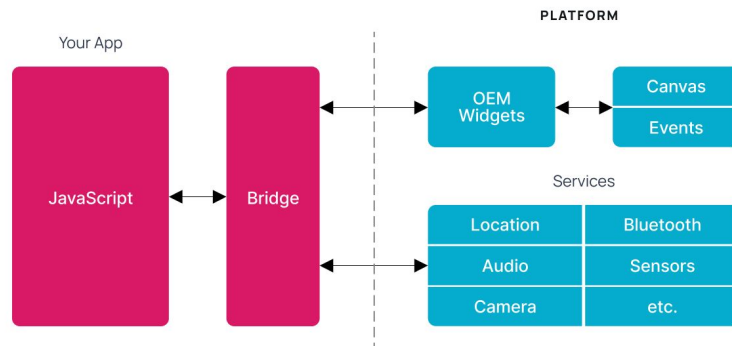
🎨 User Experience:

🔧 Functionality:

📊 Stability and Reliability:

⌚ Proof of work:

### React Native



### Flutter

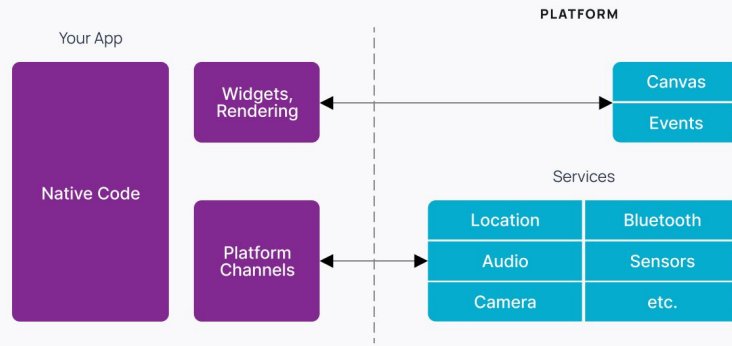


Photo Source:

<https://brocoders.com/blog/react-native-vs-flutter-which-one-better/>

# Side by side\_

## React Native & Flutter

🚀 Performance: Flutter

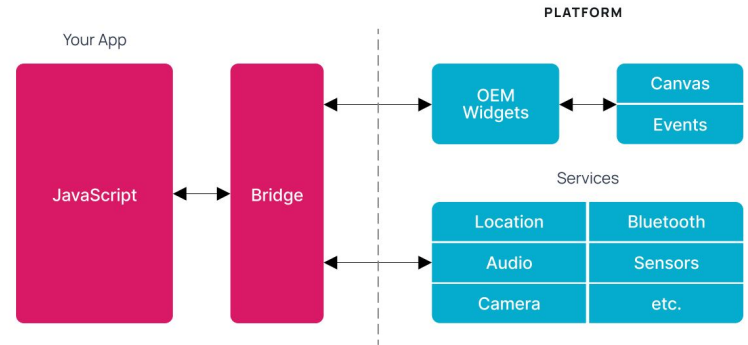
🎨 User Experience:

🔧 Functionality:

📊 Stability and Reliability:

🕒 Proof of work:

### React Native



### Flutter

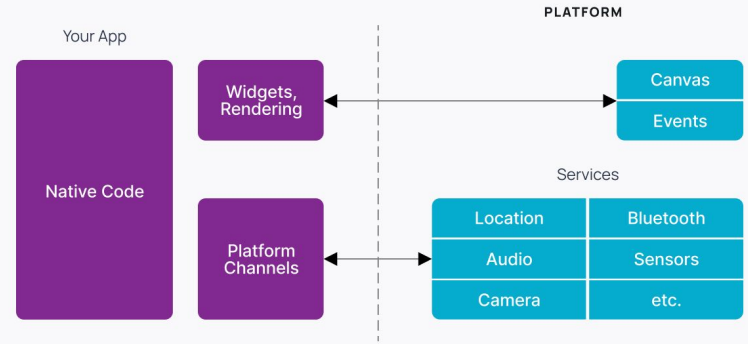


Photo Source:

<https://brocoders.com/blog/react-native-vs-flutter-which-one-better/>

# Side by side\_

## React Native & Flutter

🚀 Performance: Flutter

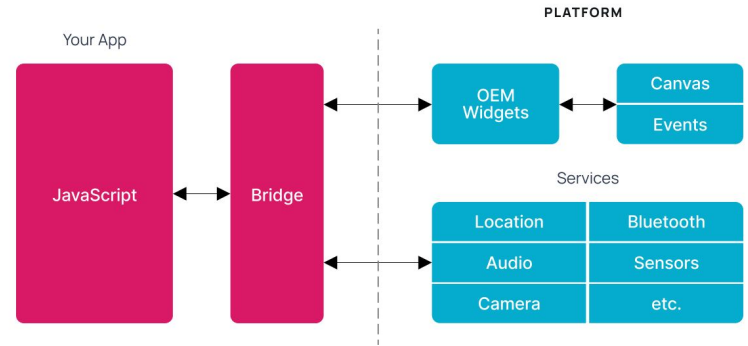
🎨 User Experience: Tie

🔧 Functionality:

📊 Stability and Reliability:

⌚ Proof of work:

### React Native



### Flutter

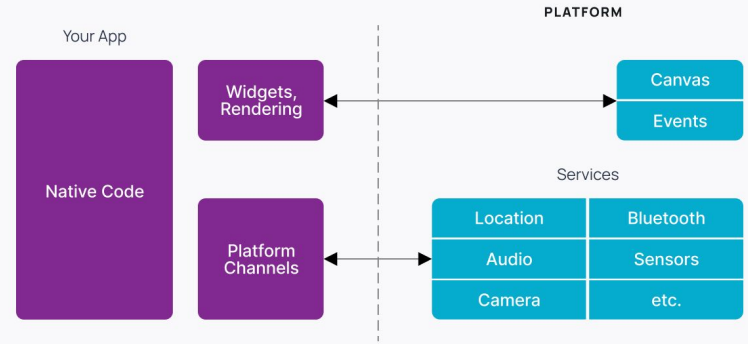


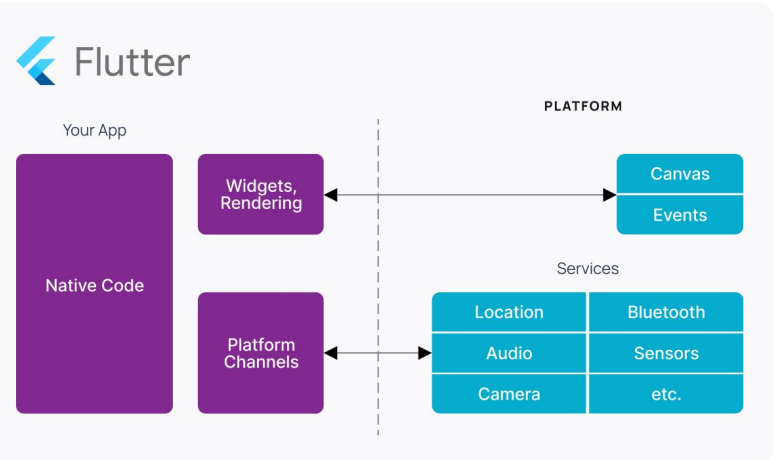
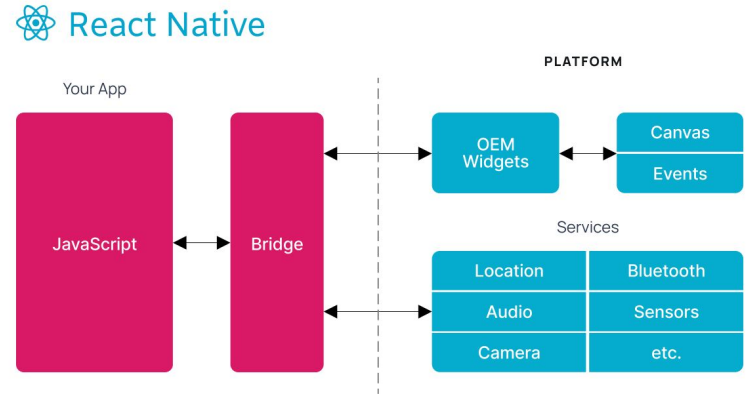
Photo Source:

<https://brocoders.com/blog/react-native-vs-flutter-which-one-better/>

# Side by side

## React Native & Flutter

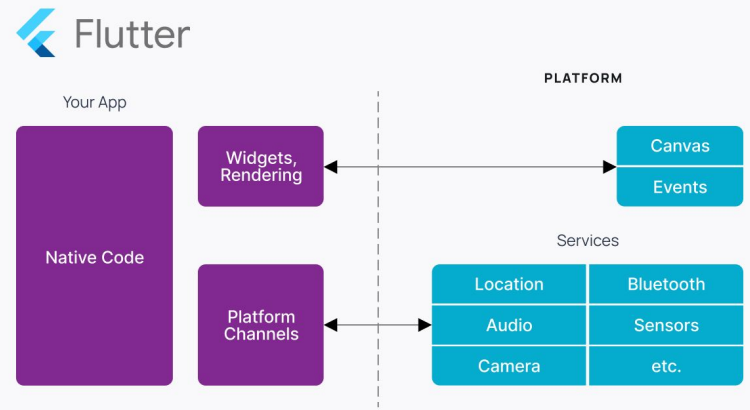
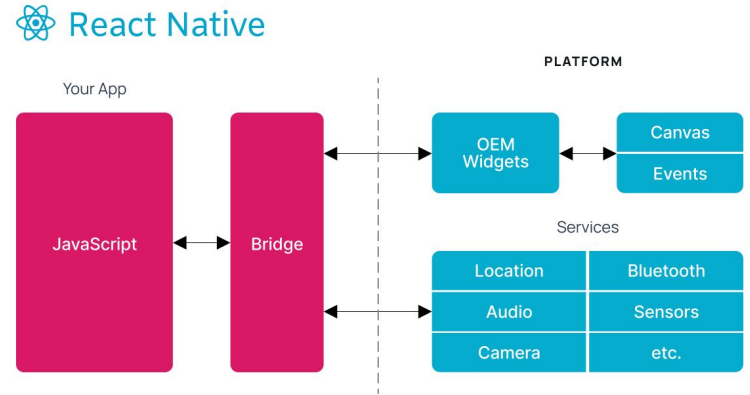
- 🚀 Performance: Flutter
- 🎨 User Experience: Tie
- 🔧 Functionality: Web vs. mobile devs
- 📊 Stability and Reliability:
- ⌚ Proof of work:



# Side by side

## React Native & Flutter

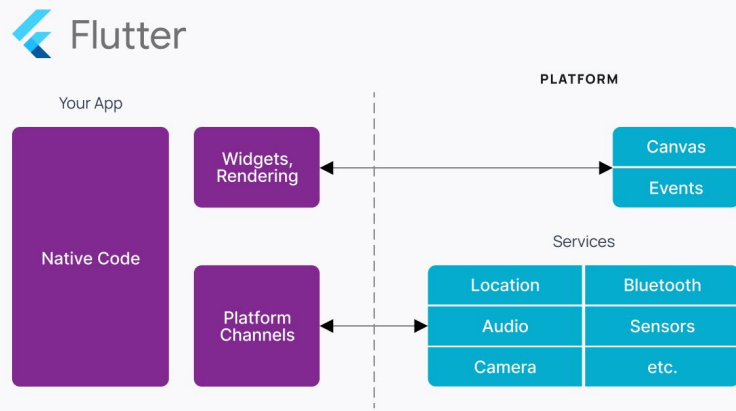
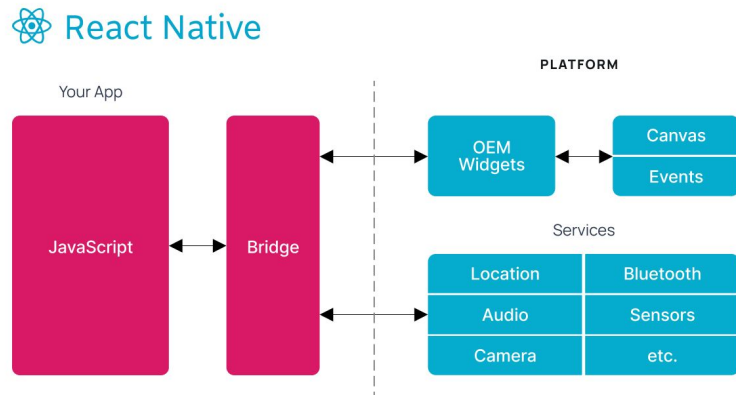
- 🚀 Performance: Flutter
- 🎨 User Experience: Tie
- 🔧 Functionality: Web vs. mobile devs
- 📊 Stability and Reliability: Flutter
- ⌚ Proof of work:



# Side by side

## React Native & Flutter

- 🚀 Performance: Flutter
- 🎨 User Experience: Tie
- 🔧 Functionality: Web vs. mobile devs
- 📊 Stability and Reliability: Flutter
- ⌚ Proof of work: React Native



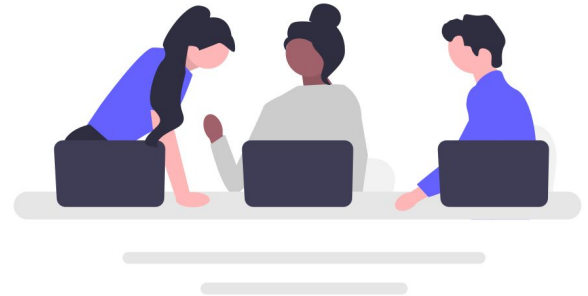
**What should  
you choose?**



# What should you choose?

Let's discuss these 3 scenarios:

- Software Developer, working for a company
- Freelancer Software Developer
- Business Owner or Leader



# Developer working for a Company\_

Things to consider

# Developer working for a Company\_

## Things to consider

### → **The amount of opportunity in the market**

In many cases, native > cross-platform at this aspect

### → **The type of company you want to work for**

Big and well established companies → native  
Startups and smaller companies → cross-platform

# Developer working for a Company\_

## Things to consider

### → The amount of opportunity in the market

In many cases, native > cross-platform at this aspect

### → The type of company you want to work for

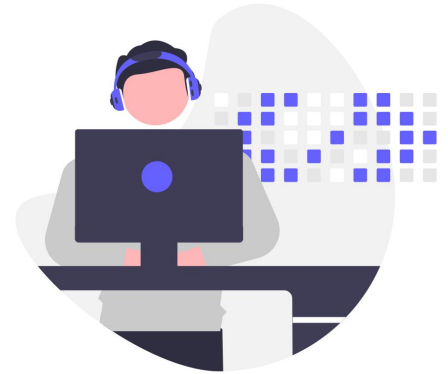
Big and well established companies → native  
Startups and smaller companies → cross-platform



Photo Source: <https://caretgrowth.com>

# Freelancer Software Developer\_

Tricky road that can pay off



# Freelancer Software Developer\_

Tricky road that can pay off

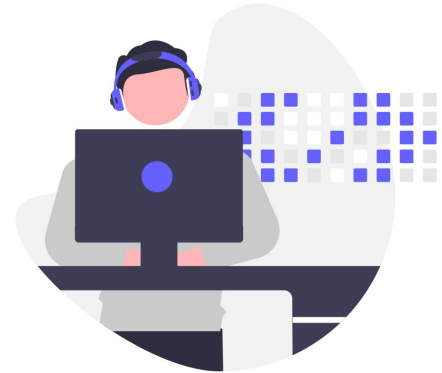
## → **Specialization vs. Versatility**

More specialized → fewer opportunities, but better paid

More versatile → plenty of opportunities, bigger competition

## → **Developer vs. Tech Consultant**

Implement yourself, or support and guide teams



# As a Business\_

Easy choice sometimes



# As a Business\_

## Easy choice sometimes

### → The “deal breakers”

IoT, Sensors, AR/VR, high stability/reliability apps (life alert)

### → The “no brainers”

E-commerce apps, data driven apps





# As a Business\_

Hard choice other times

# As a Business \_

## Hard choice other times

### → Project timeline and budget

Deadlines? Team size?

### → Strategy

Start with one OS first? Or rebuild it later?

### → Your technical capabilities

Web vs. mobile devs, learning time, ROI

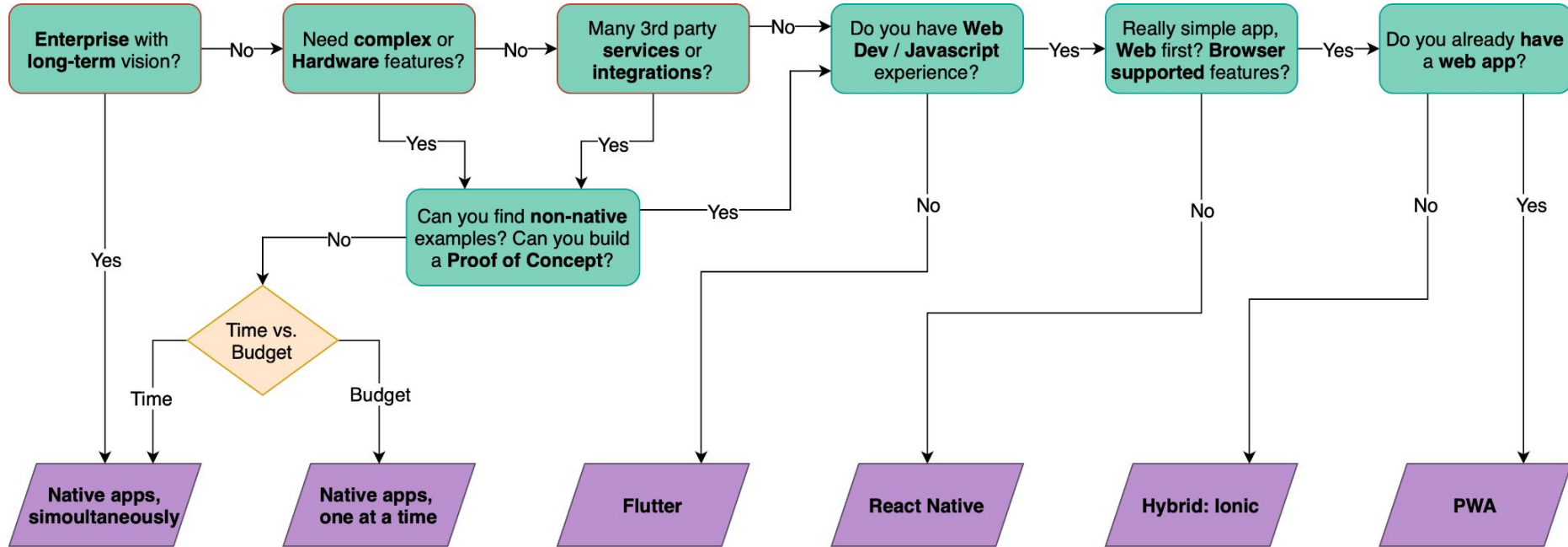


# As a Business\_

Decision chart

# As a Business

## Decision chart



# Takeaways


# Takeaways\_

 It's a **great time** to build an **app\***

 **Native, Cross-platform**, Hybrid, PWA

 The difference? **road** versus **pathway**

 **Career choice?** - options and preferences

 **Business choice?** - needs and possibilities

# Takeaways\_

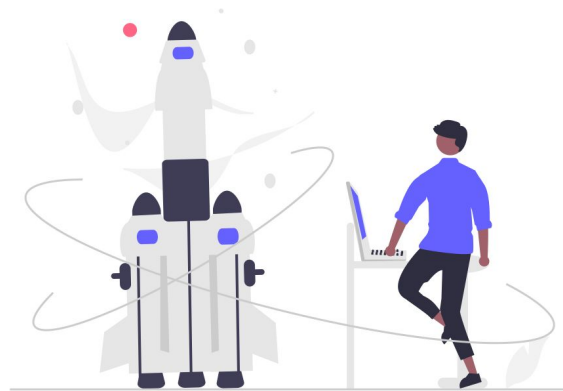
🕒 It's a **great time** to build an **app\***

📱 **Native, Cross-platform**, Hybrid, PWA

🚦 The difference? **road** versus **pathway**

👤 **Career choice?** - options and preferences

📊 **Business choice?** - needs and possibilities



# Q&A

## Ask me anything!



**Dan Ilies**

Head of Mobile Development  
@ Wolfpack Digital



Instagram: [@mobiledevlife](https://www.instagram.com/mobiledevlife)



Email: [danilies92@gmail.com](mailto:danilies92@gmail.com)