

MOBILE APP DEVELOPMENT TECHNOLOGIES

Different technologies can help you build amazing mobile apps. Here's a quick comparison to help you choose the right one for your next project.



React Native

Learn once, write anywhere



Flutter

Build beautiful apps quickly



Native Development

Maximum performance and capabilities

	React Native	Flutter	Native Development
 Platforms	 Android & iOS	 Android & iOS	 Android & iOS
 Code Sharing	 ~90% High	 ~80% High	 0% None
 Performance	 Good Near Native	 Very Good Near Native	 Excellent Native
 UI/UX	Good Customizable	Excellent Rich & Flexible UI	Excellent Platform-specific UI
 Development Speed	 Fast High productivity	 Very Fast Hot reload, rich widgets	 Slower Separate codebases
 Cost	 Cost Effective One codebase	 Cost Effective One codebase	 Higher Cost Two codebases
 Best For	Cross-platform apps with large community support	Apps with beautiful UI and fast development	Complex apps needing high performance & platform-specific features



KEY TAKEAWAY

Choose the technology based on your project needs, team skills, budget, and long-term goals.
There's no one-size-fits-all—each technology has its strengths!



NATIVE MOBILE DEVELOPMENT

Native mobile development means building apps specifically for a platform using platform-specific languages and tools to deliver the best performance, security, and user experience.



High Performance
Optimized for platform hardware and OS.



Better Security
Leverages platform security features.



Best User Experience
Native look, feel, and smooth interactions.



Full Device Access
Access to all native APIs and capabilities.



ANDROID DEVELOPMENT

Language: **Kotlin**



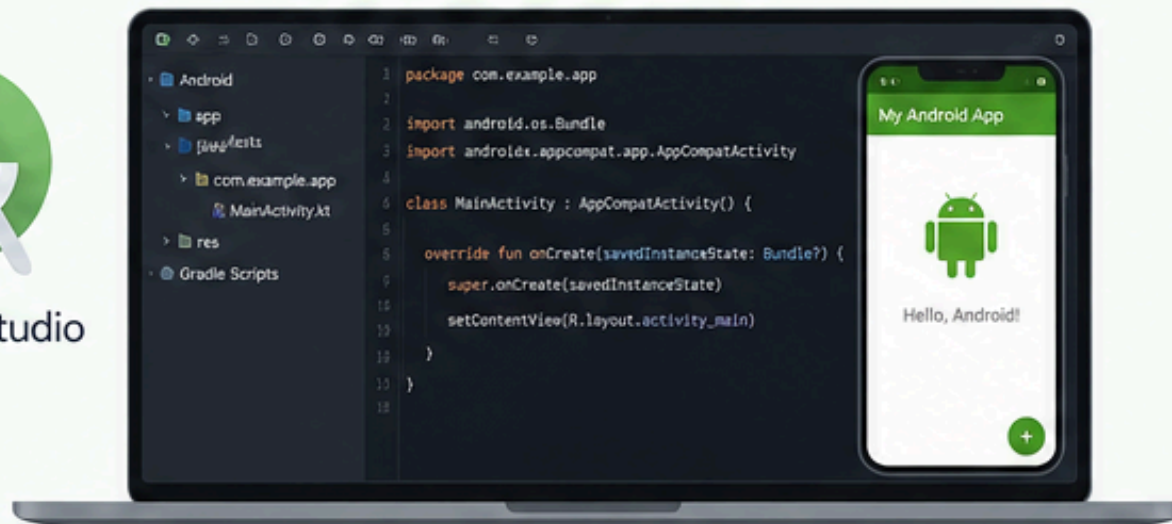
Kotlin

Kotlin is the official language for Android development. Modern, concise, and fully interoperable with Java.

IDE: **Android Studio**



Android Studio



Official IDE for Android development with powerful tools, emulator, debugger, and built-in performance profilers.

VS



iOS DEVELOPMENT

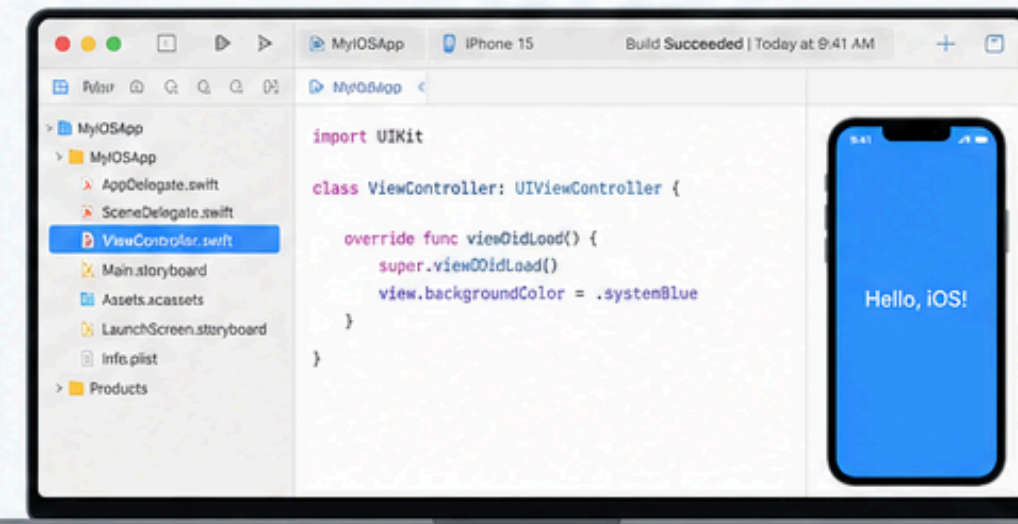
Language: **Swift**



Swift

Swift is the modern, safe, and fast programming language for iOS, iPadOS, macOS, watchOS, and tvOS.

IDE: **Xcode**



Xcode

Official IDE for Apple platforms with advanced tools, simulator, debugger, and interface builder.



NATIVE = BETTER

Native apps deliver superior performance, reliability, and user satisfaction by leveraging the full power of the platform they are built for.



NATIVE DEVELOPMENT

Built for Performance. Designed for Excellence.



HIGH PERFORMANCE

Native apps are faster, more responsive and built to deliver maximum performance.



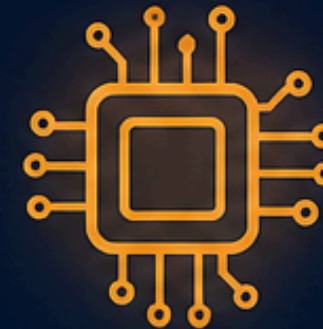
SECURITY

Leverages platform-level security features to keep your data and users protected.



SMOOTH ANIMATIONS

Delivers fluid, natural animations and a superior user experience.



HARDWARE ACCESS

Direct access to device hardware and features for powerful capabilities.



PLATFORM OPTIMIZATION

Optimized for each platform's guidelines and user expectations for the best results.



Native development delivers the **best performance, security, and user experience** by leveraging the full power of each platform.

NATIVE DEVELOPMENT DISADVANTAGES

Powerful and optimized, but comes with trade-offs.

1

SEPARATE CODEBASES FOR ANDROID & iOS



Maintain two distinct codebases for Android and iOS, causing duplication of effort.

2

HIGHER DEVELOPMENT COSTS



Building and maintaining two separate apps requires more budget across the entire development lifecycle.

3

LONGER DEVELOPMENT TIME



Developing features, fixing bugs, and releasing updates takes nearly twice the time.

4

LARGER DEVELOPMENT TEAMS



Requires more developers with platform-specific expertise, increasing team size and management complexity.



Native development delivers top performance and platform-specific capabilities, but comes with higher cost, time, and resource investment.

React Native

Build once. Run anywhere.

React Native is an open-source framework by Meta for building cross-platform mobile applications using React and native capabilities.



JavaScript



TypeScript



React



One Codebase

Write once in JavaScript or TypeScript and run on multiple platforms.



Cross-Platform

Deliver a native experience on Android and iOS.



Native Performance

Leverage native components and APIs for high performance and reliability.



Rich Ecosystem

Large community, powerful tools, and extensive libraries.

One Codebase. Multiple Platforms.

```
App.tsx
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>
        Hello, React Native! 🙌
      </Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center',
    alignItems: 'center', backgroundColor: '#f5f5f5' },
  title: { fontSize: 24, fontWeight: '600', color: '#111' },
});
```



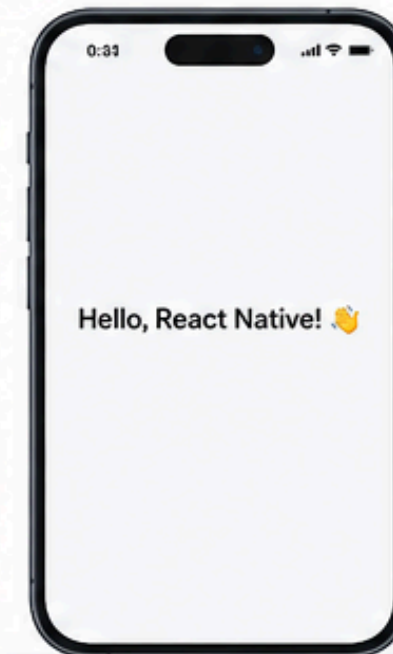
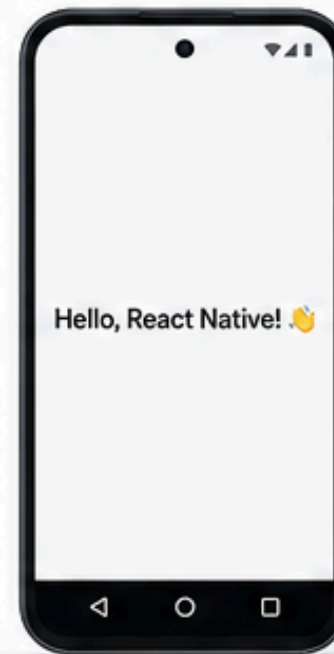
React Native

ONE CODEBASE



Android

Run on Android devices



iOS

Run on iOS devices



React Native Architecture

Learn Once, Write Anywhere



JavaScript
Cross-Platform



Bridge
Connects



Native UI
High Performance

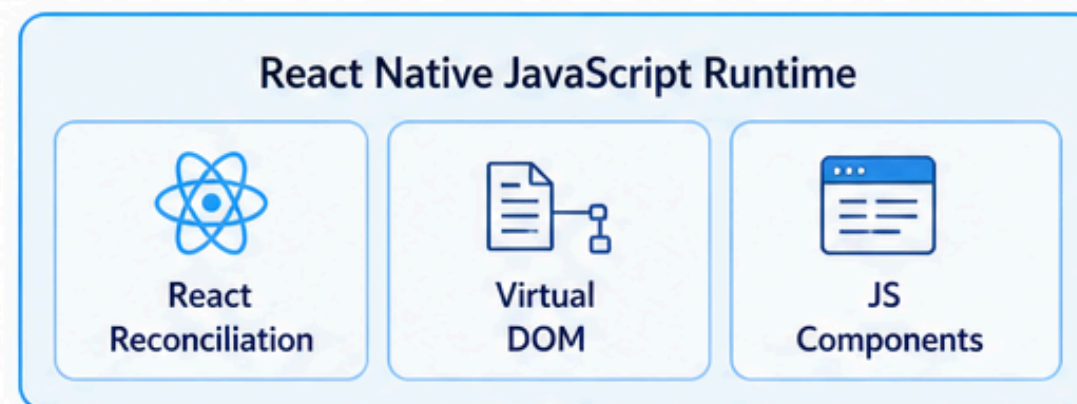
JS JavaScript (React Native)

```
1 import {
2   View,
3   Text,
4   Button,
5   NativeModules,
6 } from 'react-native';
7
8 const { DeviceInfo } = NativeModules;
9
10 export default function App() {
11   const getDeviceName = async () => {
12     const name = await DeviceInfo.getName();
13     console.log('Device:', name);
14   };
15
16   return (
17     <View style={{ flex: 1, padding: 20 }}>
18       <Text style={{ fontSize: 20 }}>
19         Hello from React Native! 🙌
20       </Text>
21       <Button title="Get Device Name"
22         onPress={getDeviceName} />
23     </View>
24   );
25 }
```

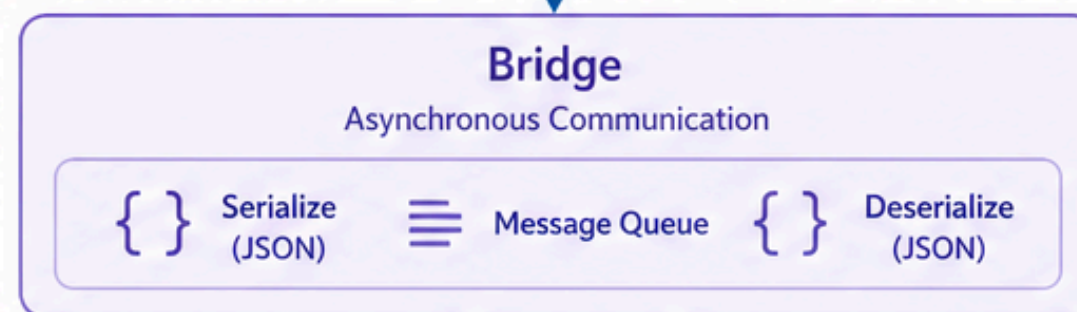
JavaScript Runs In



JavaScript Engine
(Hermes / JSC)



Batched Updates



Events
(Touch, Scroll,
Text Input, etc.)



Android Device

Invoke Native
Modules / UI
Commands



iOS Device

How It Works

- 1 JavaScript code is written using React Native and runs in a JavaScript Engine.
- 2 The JS runtime creates a virtual UI tree and calculates changes.
- 3 Updates are sent across the Bridge in batches as serialized messages.
- 4 Native side receives the commands and updates the actual UI using platform APIs.
- 5 Events from native (e.g., taps, scrolls) are sent back across the Bridge to JavaScript.



Key Benefits



Cross-Platform
One codebase for Android & iOS



Native Performance
Uses real native components



Developer Productivity
Hot Reloading, Fast Refresh



Extensible
Write Native Modules

7

How React Native Works

One codebase, two native platforms.

1. JavaScript (React Native)

```
1 import React from 'react';
2 import {
3   View,
4   Text,
5   Button,
6 } from 'react-native';
7
8 export default function App() {
9   return (
10     <View style={{ flex: 1,
11       justifyContent: 'center',
12       alignItems: 'center' }} >
13       <Text style={{ fontSize: 22 }}>
14         Hello, React Native! 🙌
15       </Text>
16       <Button title="Press Me" onPress={() =>
17         alert('Button Pressed!')
18       } />
19     </View>
20   );
21 }
```



Your App Logic

Written in JavaScript with React

JS Sends Commands

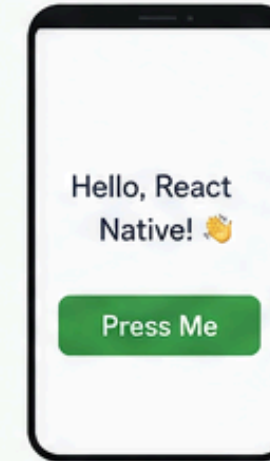


2. Bridge

The Bridge allows JavaScript to communicate with native modules asynchronously.

Native Sends Events

3A. Native (Android)



Android Native Modules



UIManager
(Render Views)

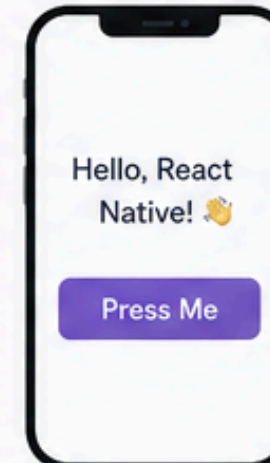


Native Modules
(Camera, Storage, etc.)



Event Emitter
(Send Events to JS)

3B. Native (iOS)



iOS Native Modules



UIManager
(Render Views)



Native Modules
(Camera, Storage, etc.)

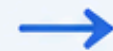


Event Emitter
(Send Events to JS)



Write Once

Build your app using JavaScript and React



Bridge

Commands are sent to native via the Bridge



Native Execution

Native modules handle UI rendering and device APIs



Events Back

Native sends events back to JavaScript



React Native combines the best of both worlds: the efficiency of native and the developer experience of JavaScript.

React Native Advantages

Build better apps, faster with React Native's powerful ecosystem and developer experience.



Single Codebase

Write once, run anywhere. Use a single JavaScript codebase for both iOS and Android.



Fast Development

Pre-built components and powerful tooling help you build and ship features much faster.



Hot Reload

See changes instantly. Hot Reload speeds up development and improves productivity.



Large Community

Backed by Meta and millions of developers worldwide. Get help, resources and updates every day.



Expo Ecosystem

Expo makes development easy with powerful tools, libraries and services out of the box.



Lower Development Cost

Faster development, code reusability and a single team for both platforms reduce overall development costs.



More value. Less time. Greater impact.



React Native

React Native Disadvantages

While powerful and efficient, React Native has some limitations to consider.



Performance Comparison

- React Native relies on a JavaScript bridge to communicate with native modules.
- This extra layer can introduce latency and reduce performance in certain scenarios.
- Native apps (Swift/Kotlin) still offer better performance for highly demanding tasks.



Native Module Dependency

- Access to device features and advanced functionalities often requires native modules.
- Some features may not be available out of the box and need custom native development.
- Dependency on third-party libraries can lead to maintenance and compatibility issues.



Challenges with Graphics-Intensive Applications

- React Native may struggle with high-end 3D graphics, complex animations, and real-time rendering.
- Not ideal for demanding use cases like high-performance games, AR/VR, or advanced video editing apps.
- Native development is better suited for such graphics-intensive applications.



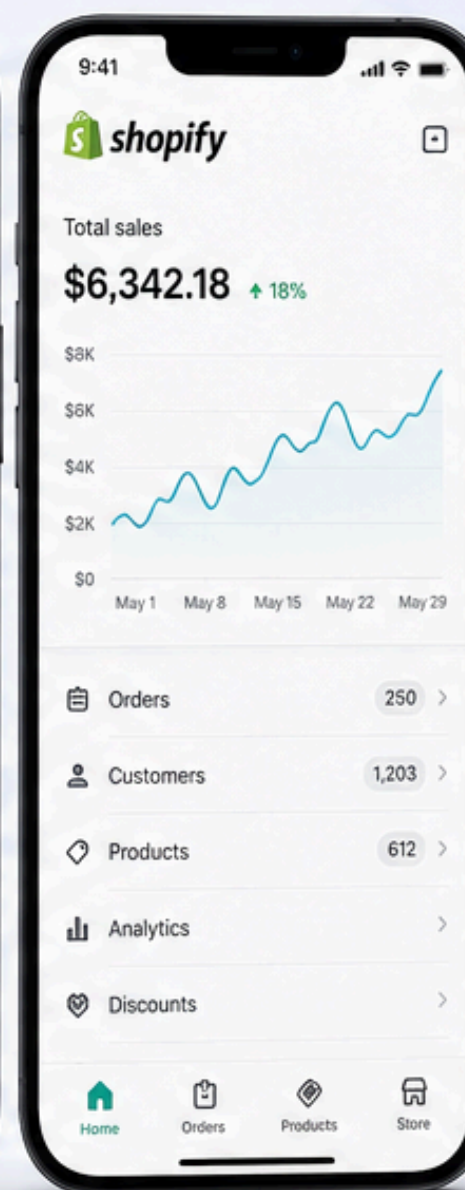
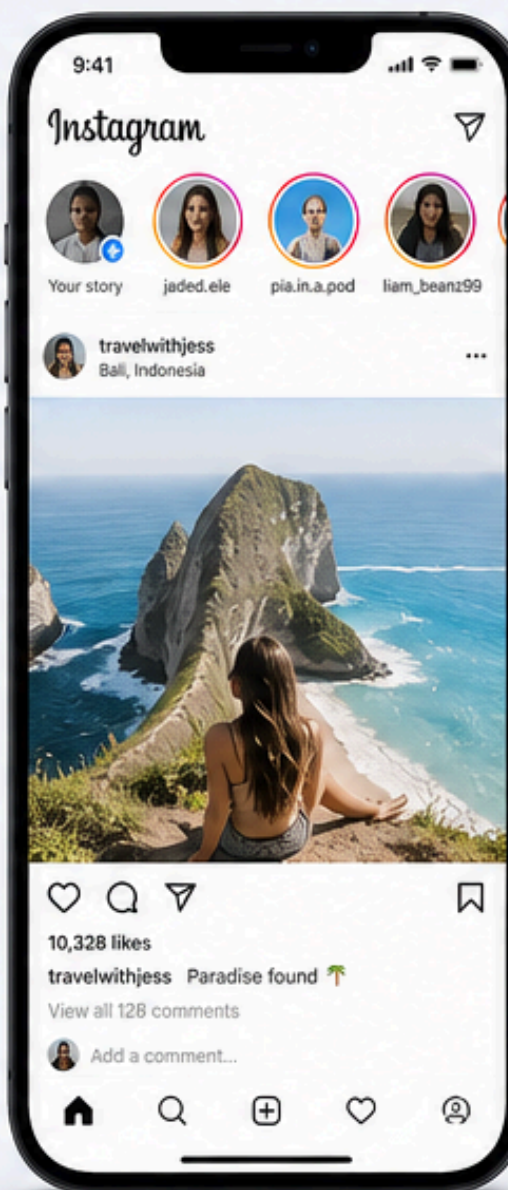
10

Popular React Native Apps

React Native powers some of the world's most popular apps, used by millions every day.



One Framework.
Limitless Possibilities.



Instagram

The world's favorite photo and video sharing app.



Facebook

Connect with friends, share updates and explore the world.



Shopify

Powering millions of businesses around the globe.



Discord

Chat, connect and build communities effortlessly.



Callstack

React Native agency app for developers.



Tesla

Control. Monitor. Drive the future.



Built with React Native. Loved by millions.

What is Flutter?

Flutter is an open-source UI toolkit by **Google** for building beautiful, natively compiled, multi-platform applications from a single codebase using **Dart**.



Dart

Flutter is built with Dart, a modern, fast and expressive programming language.



Flutter

A portable UI toolkit for building beautiful, fast, and consistent apps across platforms.



Write Once. Run Anywhere.

Build for multiple platforms from a single codebase.

```
main.dart
import 'package:flutter/material.dart';
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(child: Text('Hello Flutter!')),
      ),
    );
  }
}
```



Flutter

Single Codebase



Android

Build for Android



iOS

Build for iOS



Web

Build for Web



Desktop

Build for Windows, macOS, Linux



Fast Development

Hot reload for rapid iteration



Beautiful UI

Rich widgets and customizable designs



High Performance

Native performance with compiled engine



Large Community

Backed by Google and a growing ecosystem

How Flutter Works

Flutter builds beautiful UI by combining Dart, a rich widget system, and the Skia graphics engine to render everything on the screen.



1. Write in Dart

Developers write Flutter apps using Dart language.



2. Widgets

Flutter uses a rich set of composable widgets to build the UI as a tree.



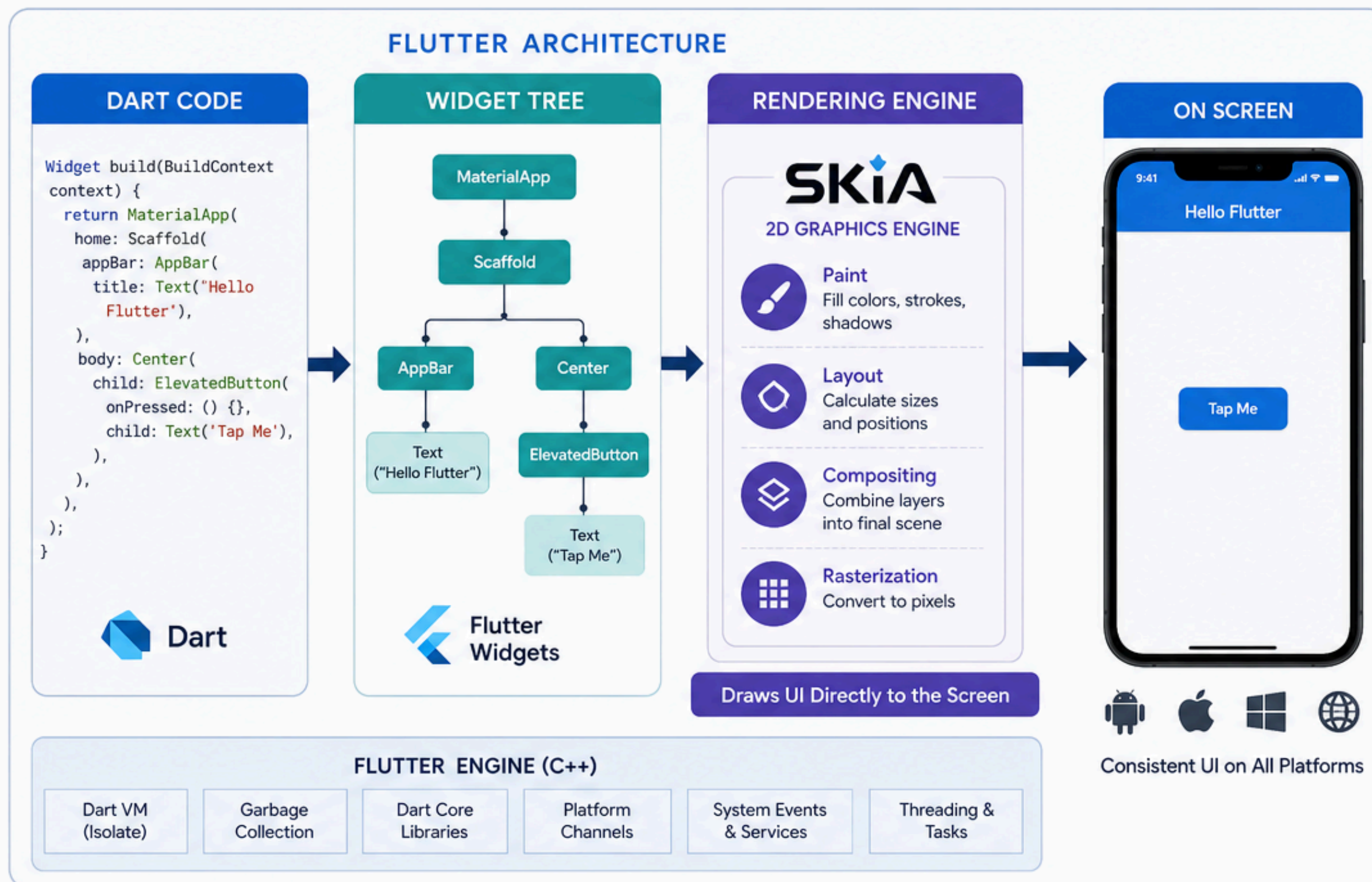
3. Rendering with Skia

Skia, the 2D graphics engine, renders every pixel. Flutter draws UI directly—no OEM widgets, no HTML.



4. Display on Screen

The pixels are displayed on the screen consistently across platforms.



One codebase. Any platform. Pixel-perfect.



High Performance



Fast Development



Expressive & Flexible UI

Flutter Advantages

Build beautiful, fast and reliable apps for multiple platforms with a **single codebase**.



Beautiful UI

Rich set of customizable widgets and modern design capabilities to build stunning, pixel-perfect interfaces.



Fast Rendering

Powered by the Skia engine and Dart, Flutter delivers lightning-fast performance and smooth user experiences.



Smooth Animations

Built-in animated widgets and transitions make it easy to create fluid, engaging animations.



Consistent Design

One codebase, one design system. Deliver a consistent look and feel across all platforms.



Cross-Platform Support

Deploy to Android, iOS, Web, Windows, macOS and Linux from a single codebase.



One Codebase. Endless Possibilities.



Android



iOS



Web



Windows



macOS



Linux

Flutter Disadvantages



1. Dart Learning Curve

Flutter uses Dart, a language that is unfamiliar to many developers, requiring time and effort to learn.



2. Larger App Size

Flutter apps are generally larger compared to native apps due to the Flutter engine and Dart runtime.



3. Ecosystem Limitations

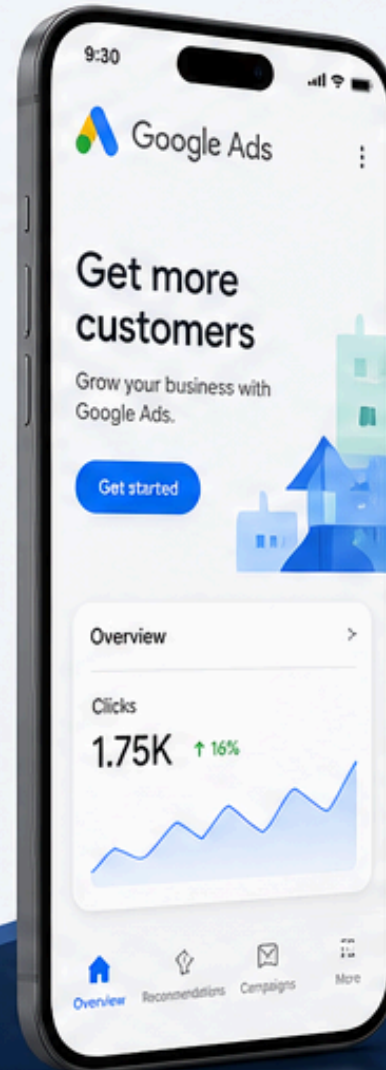
Smaller package ecosystem and fewer third-party plugins compared to more established frameworks.

Popular Flutter Apps

Flutter powers some of the world's leading apps across industries. Here are a few popular examples built with Flutter.

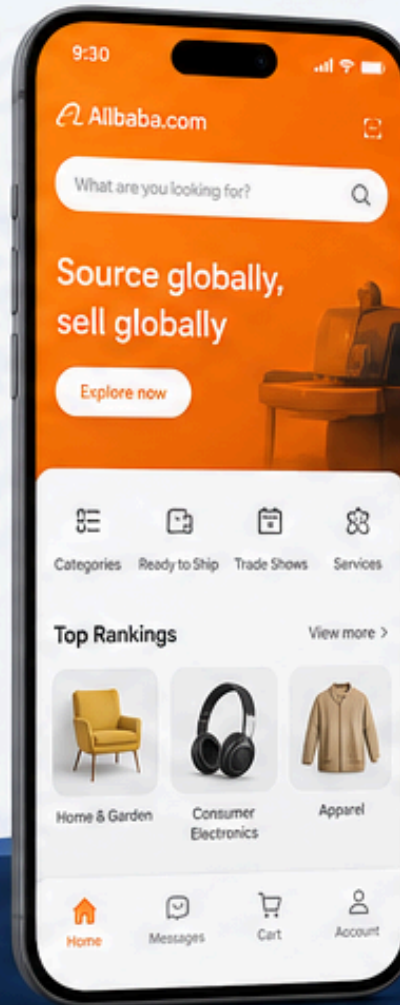


One Framework.
Limitless Possibilities.



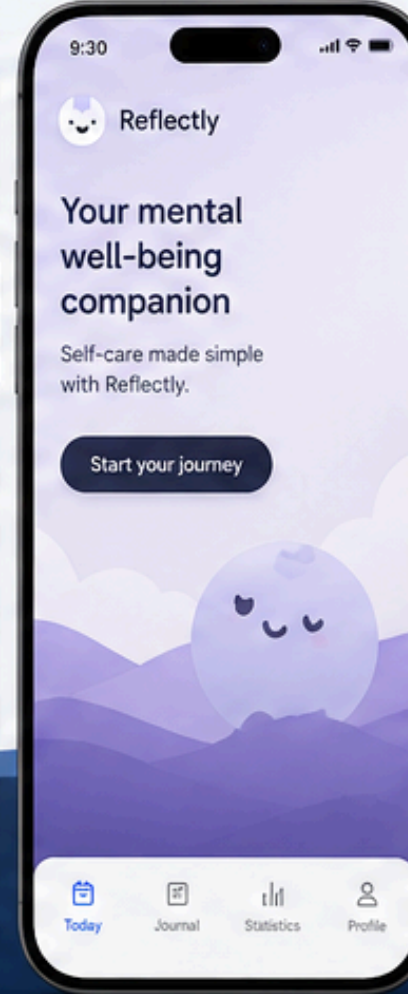
Google Ads

Reach more customers and grow your business with ease.



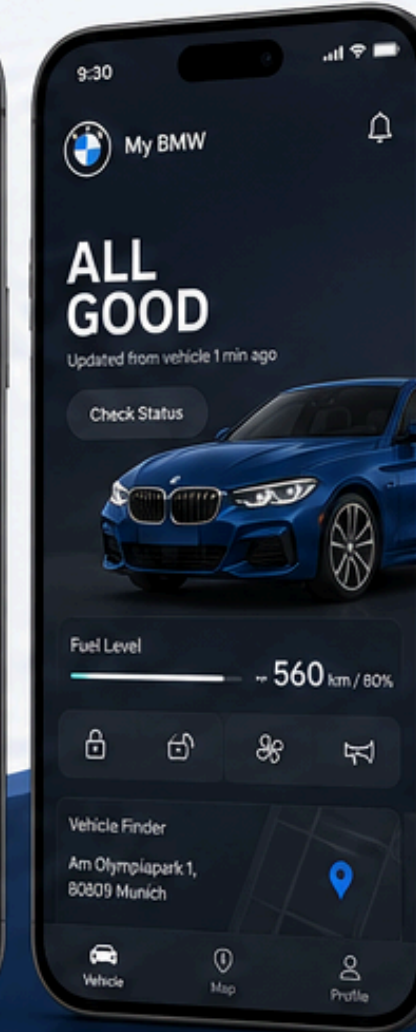
Alibaba

Global B2B marketplace connecting millions of buyers and suppliers.



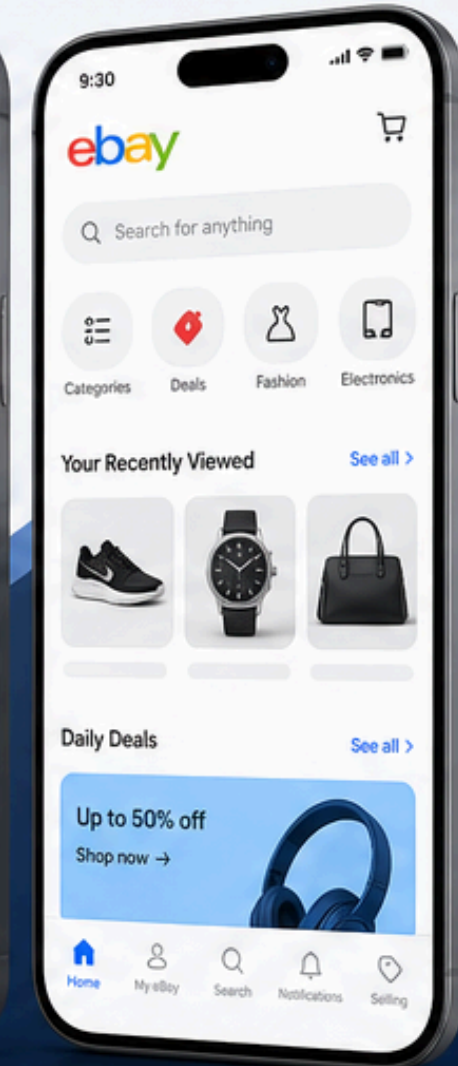
Reflectly

AI-powered journal that helps you build better habits.



BMW

Connected driving experience with real-time vehicle insights.



eBay

Shop, sell and discover millions of items from around the world.



Beautiful UI. Native Performance. One Codebase.

PERFORMANCE COMPARISON

Native vs Flutter vs React Native



Native delivers the highest raw performance.

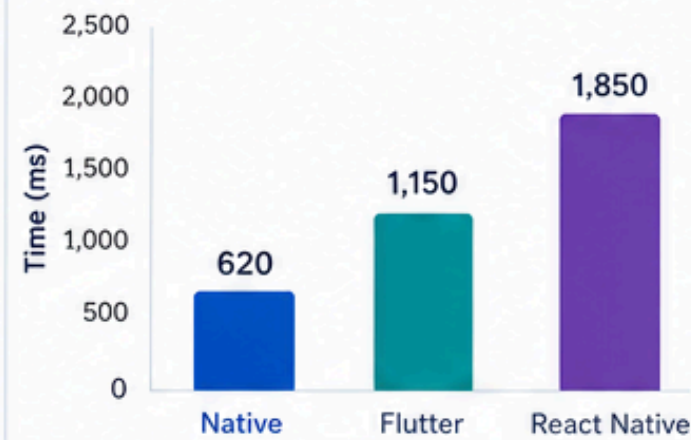
Flutter comes close with excellent consistency.

React Native shows strong performance with room to optimize.

Native (Kotlin/Swift) **Flutter** (Dart) **React Native** (JavaScript/TS)

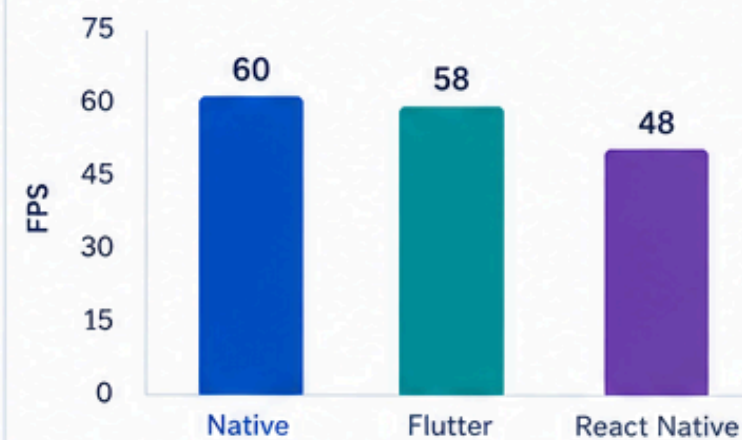
APP STARTUP TIME (COLD START)

Lower is Better



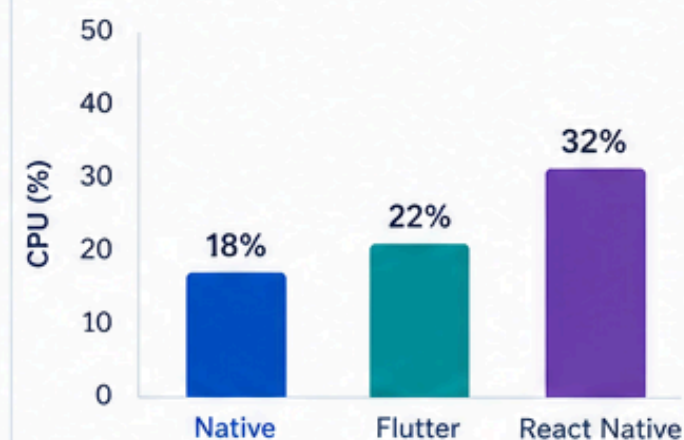
UI RENDERING (AVERAGE FPS)

Higher is Better



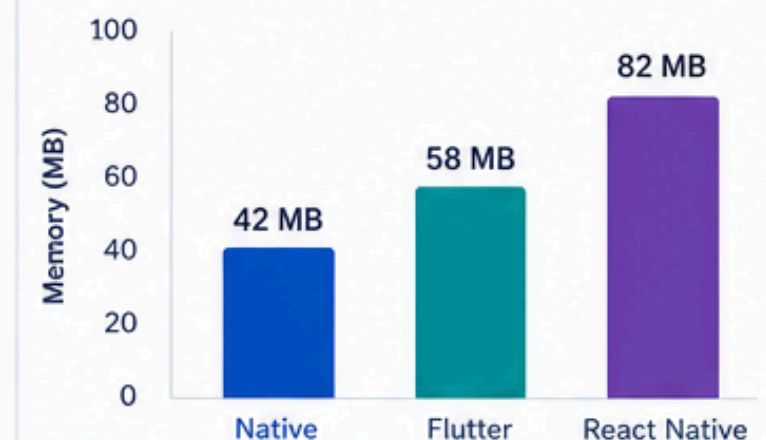
CPU USAGE (TYPICAL APP)

Lower is Better



MEMORY USAGE (AVG)

Lower is Better



OVERALL PERFORMANCE SCORE

Higher is Better

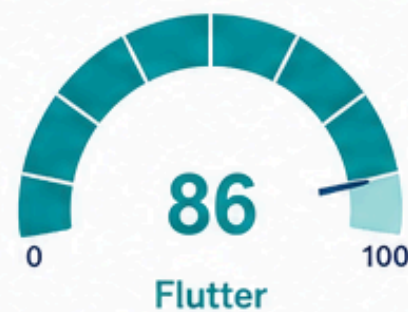


Best raw performance across all metrics



OVERALL PERFORMANCE SCORE

Higher is Better



Excellent balance of performance & productivity



OVERALL PERFORMANCE SCORE

Higher is Better



Good performance with optimization potential



DETAILED COMPARISON

	Native (Kotlin/Swift)	Flutter (Dart)	React Native (JavaScript/TS)
Startup Time (Cold Start)	620 ms (Best)	1,150 ms (Good)	1,850 ms (Average)
UI Rendering (Average FPS)	60 FPS (Best)	58 FPS (Excellent)	48 FPS (Good)
CPU Usage (Typical App)	18% (Best)	22% (Good)	32% (Average)
Memory Usage (Average)	42 MB (Best)	58 MB (Good)	82 MB (Average)
Battery Impact (Typical App)	Low (Best)	Low (Good)	Medium (Average)
Native Performance (Access)	Full Access (Best)	Near Native (Excellent)	Bridge Overhead (Average)



KEY TAKEAWAY

Native leads in raw performance, Flutter is a very close second with consistent results, and React Native offers solid performance with continuous improvements.

Tested on mid-range Android device (Snapdragon 778G) & iPhone 13
Results may vary based on app complexity and implementation

17

Development Speed Comparison

Build faster. Deliver sooner.



Faster Development

Shorter time from start to finish



Higher Coding Efficiency

More done in less time



Better Project Completion

More projects delivered on time



NATIVE

Swift (iOS) /
Kotlin (Android)

PROJECT TIMELINE

24–28
Weeks



CODING EFFICIENCY

40%

Baseline (Single Codebase: No)

PROJECT COMPLETION RATE

60%

Projects completed
on time

Lower



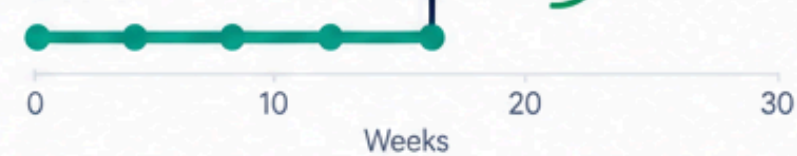
FLUTTER

Dart

★ FASTEST OVERALL

PROJECT TIMELINE

12–16
Weeks



40–50%
FASTER

CODING EFFICIENCY

80%

Single Codebase for iOS & Android

PROJECT COMPLETION RATE

90%

Projects completed
on time

Highest

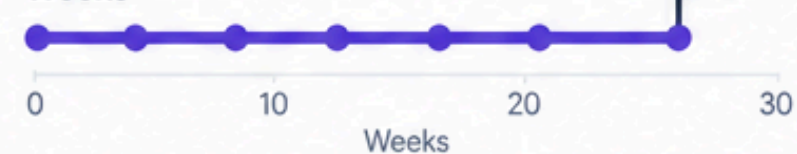


REACT NATIVE

JavaScript /
TypeScript

PROJECT TIMELINE

16–20
Weeks



25–30%
FASTER

CODING EFFICIENCY

70%

Single Codebase for iOS & Android

PROJECT COMPLETION RATE

75%

Projects completed
on time

Good



Key Takeaway

Flutter leads in development speed, coding efficiency, and on-time project delivery—helping you **build more, faster**.



Save Time

Ship features faster



Write Less Code

Do more with less



Deliver More

Higher success rate

Cost Comparison

Comparing development costs across Native, Flutter, and React Native based on team size, budget estimation, and resource requirements.



Key Takeaway

Cross-platform frameworks significantly reduce development time and cost while maintaining high performance.

Criteria	 Native (Swift – iOS / Kotlin – Android)	 Flutter	 React Native
 Team Size (Typical)	 6 – 10+ (2 Platforms)	 3 – 5 (1 Team)	 3 – 5 (1 Team)
 Budget Estimation (Relative Range)	\$\$\$ High (100%)	\$\$ Moderate (60 – 70% of Native)	\$\$ Moderate (60 – 70% of Native)
 Development Time (Relative)	 9 – 12+ Months (Longest)	 5 – 7 Months (30 – 40% Faster)	 6 – 8 Months (20 – 30% Faster)
 Resource Requirements	• Separate iOS & Android Teams • Platform-specific Developers • More QA & DevOps Effort	• Single Codebase • Flutter Developers • Fewer Resources Needed	• Single Codebase • JavaScript/TypeScript Developers • Fewer Resources Needed
 Maintenance Cost (Long Term)	 High (Two codebases to maintain)	 Lower (Single codebase)	 Lower (Single codebase)



Overall Insight

Flutter and React Native can reduce development costs by up to 30–40% compared to Native, with faster time-to-market and optimized resources.



* Percentages are indicative and may vary based on project complexity, team location, and business requirements.

Which One Should You Choose?

Different goals. Different needs. Here's how to decide.



Pro Tip: You can always start simple and evolve your stack as your goals grow.



Why React Native for This Course?

React Native is a future-ready framework that empowers you to build real-world mobile apps, integrate AI, and grow your career or startup with confidence.



TypeScript First

Build robust, scalable, and type-safe applications with modern JavaScript + TypeScript.



Expo Advantage

Faster development with Expo CLI, Go, and powerful built-in tools & libraries.



AI Integration Ready

Easily integrate AI features and APIs to build smarter, intelligent mobile apps.



Cross-Platform Power

Write once, run anywhere. Deliver native-like experiences on Android & iOS.



Freelancing Opportunities

High demand in the job market. Build in-demand skills and earn as a React Native freelancer.

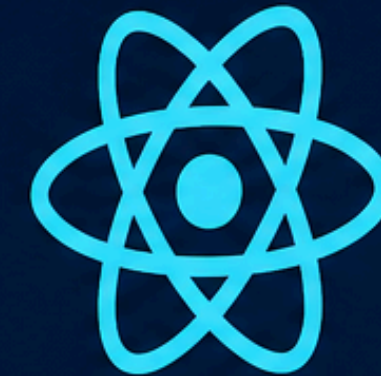


Perfect for Startups

MVPs, prototypes, and full apps—ship faster, validate ideas, and scale your startup efficiently.



This course is designed to make you **job-ready, freelance-ready, and startup-ready** with the most in-demand mobile development stack.



React Native

Learn Once.
Build Everywhere.



21

Final Recommendation

Choose the right path. Build the future.



React Native

- ✓ Cross-platform
- ✓ Large ecosystem
- ✓ Great for startups & teams

Flexible • Fast • JavaScript/TypeScript



Flutter

- ✓ High performance
- ✓ Beautiful UI
- ✓ Strong tooling

Powerful • Consistent • Dart



Native (Swift/Kotlin)

- ✓ Maximum performance
- ✓ Full platform capabilities
- ✓ Best for complex apps

Native Power • Platform First



RECOMMENDED STACK

React Native + TypeScript + Expo + AI

The most beginner-friendly, future-proof stack in 2026.



Type Safety
Fewer bugs, better code



Hot Reload
Instant feedback loop



Expo Power
Fast setup, easy deployment



Huge Community
Always support when you need it



AI Assistance
Code faster, learn smarter



Future Ready
Scalable, flexible, modern

YOUR ROADMAP TO SUCCESS



1

Learn

Master the basics
(JS/TS & React)



2

Build

Create small projects
Build consistency



3

Ship

Build real apps
with Expo



4

Scale

Optimize, add features
and grow



5

Lead

Build, share, and
inspire others

“

2026 is **YOUR** year to build.
Start today. Ship tomorrow.



Keep Learning



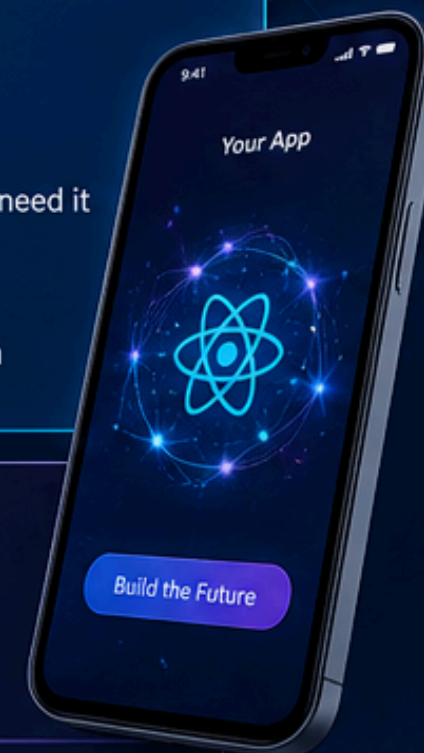
Keep Building



Keep Growing



The best time to start was yesterday.
The next best time is **NOW**.



Stay curious. Be consistent. Build amazing things.

React Native + TypeScript + Expo + AI = Your Superpower in 2026 ⚡



Thank You



Questions & Answers

