Different Approaches To Mobile App Development
There are four main approaches:

1. Native app development
2. Hybrid app development
3. Compiled app development
4. Web app development

Proponents of native solutions claim that the cross-platform approach is more faulty, provides an inferior experience, and could never replace apps written with Objective-C and Swift for iOS, and Java and Kotlin for Android. However, if you choose the right tools, these claims are no longer valid and the notion of abandoning the idea of native development makes perfect sense for businesses of all sizes. We will go in detail of the four approaches below.

1. Native App Development
Native apps are built for a specific platform and with the tools for this platform. iOS and Android platforms also have a different mobile application design. Android requires coding in Java or Kotlin, using Android Studio for the environment. iOS requires coding in Objective-C or Swift and the IDE is Xcode. Therefore, this approach requires at least two developers or even development teams to build the two versions of one application. If a browser version is required, then a third version of the application needs to be implemented.

Pros	Cons
High functionality, reliability, and performance	Source code only works on the targeted platform
Easiest integration of the advanced features (AR/VR, face recognition, machine learning)	Developers need to know each of the platforms' languages
Runs inside the operating system	Doubled time and price of development
Fine-tuned to run on a specific platform	Slower to market due to multiple source codes
Ready access to device utilities (can use the APIs like Camera, audio, network, storage, GPS, Bluetooth, NFC, etc)	

2. Hybrid App Development

A hybrid app is a web app built with HTML, CSS, and JavaScript that is wrapped by a special browser UI component WebView in Android or UIWebView in iOS. It has the ability to transfer native code calls to the web application, as well, as dispatch JavaScript messages back to the native part of the mobile app. So, a web app, which is 'wrapped' by native code gets to access the device's hardware resources. The result is a real native app that can be downloaded from the app store. They're time- and cost-effective.

Pros	Cons
Easy to build	Dependent on middleware (PhoneGap, Ionic, etc)
Much cheaper than a native app	Middleware may be slow to update
Single app for all platforms	Some bug fixes require middleware updates
Can usually access device utilities using an API	Some bug fixes are outside of your control
Faster to develop than native apps	Slower performance
Single source code	More issues from devices fragmentation
Access to all platform APIs	High regression testing costs – a shared code base means making a change that works on one platform may introduce bugs for another platform
Web portions can update on the fly	

Examples of apps include: Dow Jones MarketWatch

3. Compiled App Development
Native cross-platform apps are created when application programming interfaces (APIs) are used that are provided by the native software development kit (SDK) but then implementation is done in other programming languages that are not the standard way of developing for a particular porting system (e.g. Swift for iOS and Java for Android). Generally, a third-party vendor provides an integrated development environment (IDE), which controls the process of producing the native app bundle for iOS and Android from a single cross-platform codebase. NativeScript, Xamarin, and React Native are the most common examples of native cross-platform languages. The final product is an app that still utilises native APIs and can achieve near-native performance, without any lag visible to the user. As such, native cross-platform app development frameworks can deliver a feature-rich, scalable and high-performance app solution.

Pros	**Cons**
Write code once and use it everywhere	You need to style components yourself
Fairly good performance	Not a good fit for apps with high-technology like AR/VR
Access to native device features	
NativeScript and React Native are open source + growing community	
Native UI components, UX, and performance	

Examples of apps include: Facebook (React Native), Mijn Inkomen Later (NativeScript), Alaska Airlines - Travel (Xamarin)

4. Progressive Web Apps (PWA)
A more lightweight alternative to Hybrid Apps — these are also regular Web apps, but instead of relying on third-party platforms and plug-ins for access to native features, they make use of modern browser APIs. The thing about PWA is that those apps are not visible in app stores, so choosing this approach will lose you this specific traffic channel. On the other hand, PWA doesn't leave the Web, so you can easily share it via URL. Progressive Web Apps are discoverable using Search Engines, and when a user gets to your site which has PWAs capabilities, the browser in combination with the device asks the user if they want to install the app to the home screen. This is huge because regular SEO can apply to your PWA, leading to much less reliance on paid acquisition. Not being in the App Store means you don't need the Apple or Google approval to be in the users pockets, and you can release updates when you want, without having to go through the standard approval process which is typical of iOS apps.

PWAs are basically HTML5 applications / responsive websites on steroids, with some key technologies that were recently introduced that make some of the key features possible. If you remember the original iPhone came without the option to develop native apps. Developers then were told to develop HTML5 mobile apps, that could be installed to the home screen, but the tech back then was not ready for this.

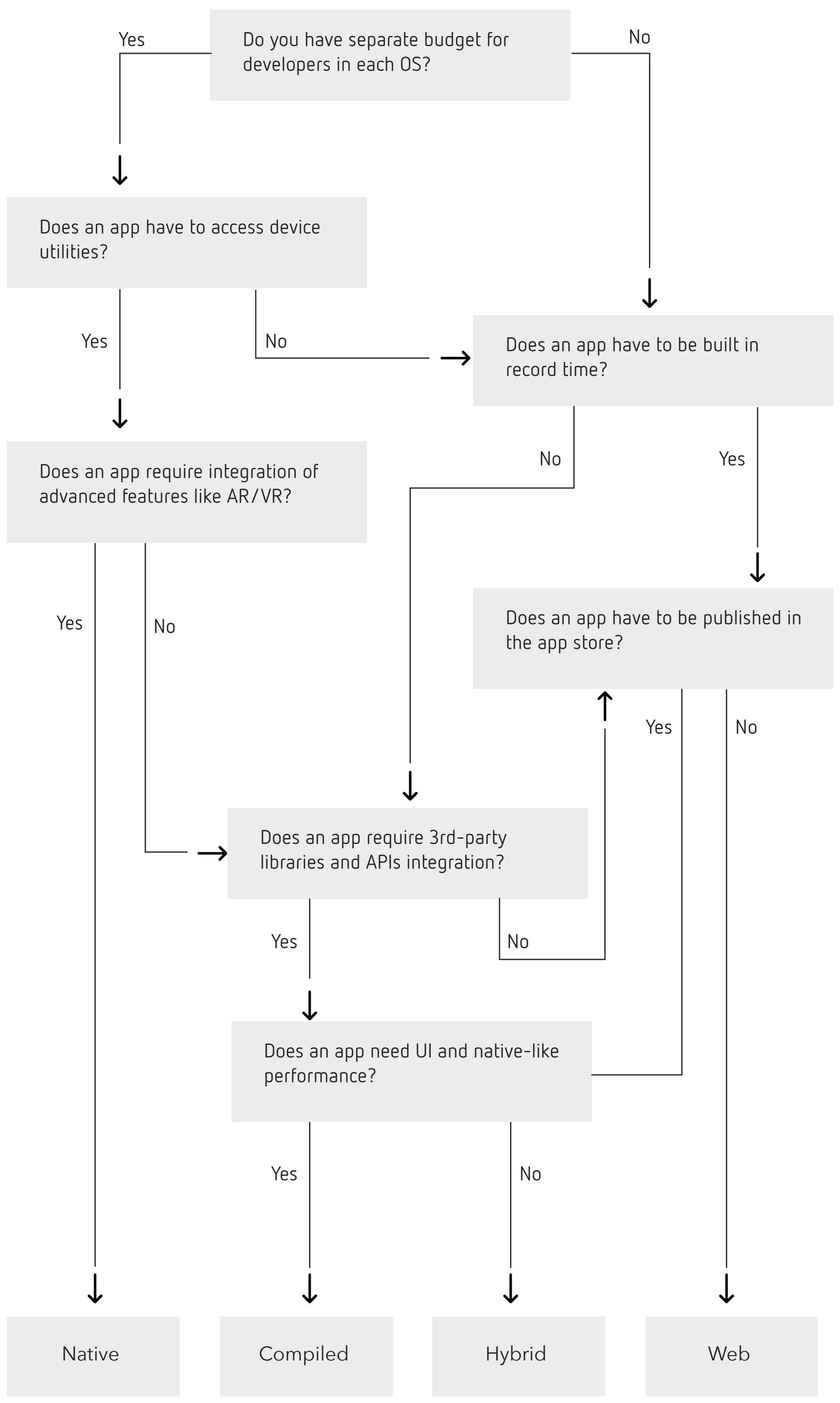
Progressive Web Apps run offline.

Support for PWA is currently strongest on Android/Chrome, while the iOS/Safari user experience is still sub-par.

Pros	**Cons**
Cross-platform	Not available in the app stores
Single code base	Need to run in a browser
Fast to production	Slower than native apps
Lower development cost	No icon on desktop
Cheaper than native and hybrid apps	Often can not access device utilities
You have both a website and an app for the price of a website	

Examples of apps include: Aliexpress

Decision Tree


Do you have separate budget for developers in each OS?
Yes
No
Does an app have to access device utilities?
Yes
No
Does an app have to be built in record time?
Does an app require integration of advanced features like AR/VR?
No
Yes
Does an app have to be published in the app store?
Yes
No
Yes
No
Does an app require 3rd-party libraries and APIs integration?
Yes
No
Does an app need UI and native-like performance?
Yes
No
Native
Compiled
Hybrid
Web