University of the West of England
BRISTOL

Week 9: Cross Platform Apps and HarmonyOS

UFCF7H-15-3 Mobile Applications

Dr Kun Wei

Want to build to both?

- iOS apps are traditionally written in Objective-C or Swift, Android apps in Java or Kotlin.
- This means if you want to build to both you may need multiple programming languages.
- Apps use the web meaning we can add HTML + CSS + JavaScript.
- Also business logic (backend) potentially Python, Ruby or GoLang.
- This is a lot for a single developer to enter the world of mobile development.

How to achieve it?

Essentially three ways:

1. Build natively on both platforms using each companies' tools.

2. Use an API, which acts as an intermediary between the code and each operating system. These mostly work through third party IDE's.

3. Use a hybrid approach which contains both mobile and web development techniques.
   - HTML5 and CSS to code the graphic UI and wrap it inside a WebView container.
   - WebViews allow developers to manipulate use some of the functionalities of the hardware (camera etc.).

Great for app studios and active developers and allows you to specialise.

Much more common and a growing area for mobile dev. Apps can achieve near native performance.

Bad for mobile dev in complex apps, as CSS and HTML5 can eat up a load of resources using battery life up fast.

What is a hybrid app?

Hybrid apps blend web elements with mobile ones created using standard web tech (HTML, CSS, JavaScript), then they are wrapped inside a native container such as a WebView.

Content in a Web view is rendered as a simple website which is responsible for

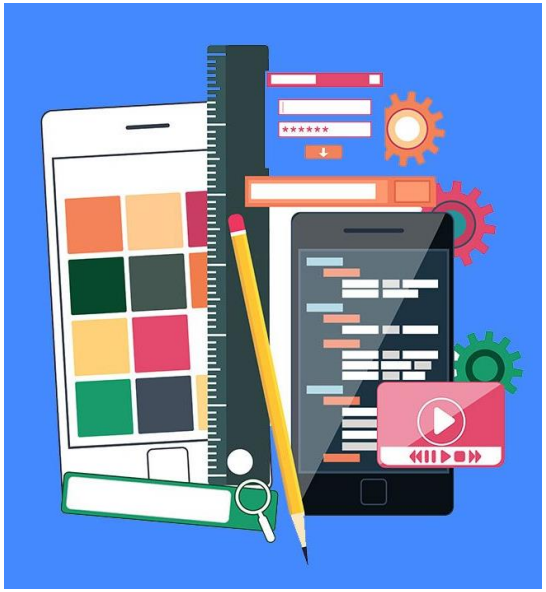
- UX
- Access to (some) hardware functions (camera, GPS, etc.).

The latter, however, can be quite limited in contexts of hybrid application development vs. native application development.

"The biggest mistake we've made as a company is betting on HTML5 over native."
Mark Zuckerberg (2012)

What is a Cross-Platform app?

- Cross-platform solutions aim to simplify (parts of) the app development process by using a single universal programming language for multiple platforms.
- Apps that are developed through a “shared operating environment”, such as Xamarin or React Native are considered “Cross-platform”.
- They offer a universal solution that is supported by several mobile platforms at once.



The good…

- Maintenance:
  - Reduce in cost when building apps and ongoing updates are easier to manage. This is very attractive to clients.
- One team:
  - One team in cheaper than two! Knowledge is shared between all members of a dev team making project management easier.
- One language:
  - Great as a solo dev or as small team. No need to worry about different syntax and approaches to development.
- Shared business logic:
  - You write less for your business logic (backend) as it only has to communicate with one front end (both Google and Apple support this regardless).

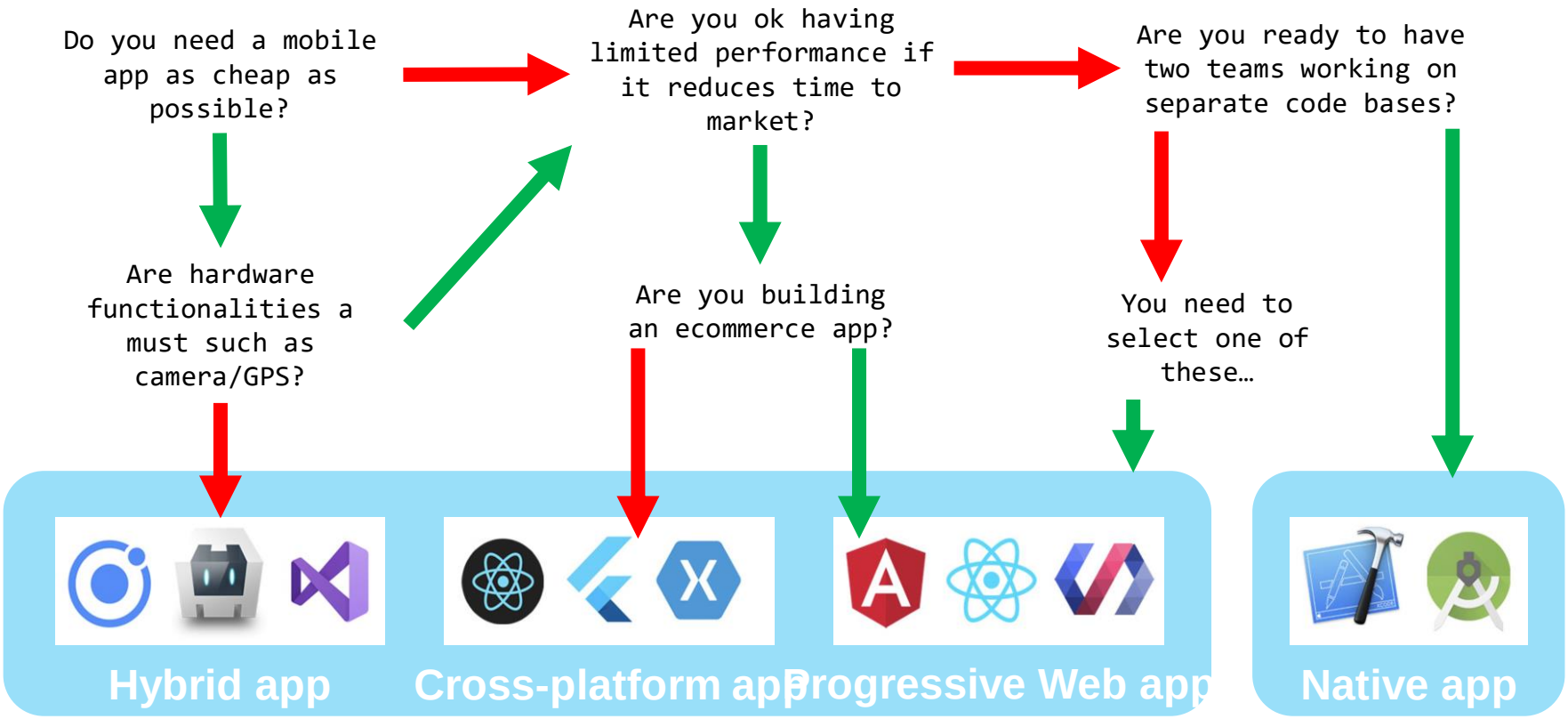
The bad…

- Different platforms…are different:
  - You can't build for one platform and expect it to be received the same on the other platform.
- Performance:
  - Faster performance than native is simply not possible. Always overhead with cross-platform (think animation smoothness or responsive UI).
- Missing new features:
  - Slow integration of new tech and features that first to release natively and are guaranteed with 100% access on each OS's platform.
- Documentation:
  - Native documentation is (often said to be) better. There are also larger online communities, so you (may) have to work harder to find support for cross-platform dev.

Worse…?

- Native is a safer bet – whole businesses depend on it:
  - Apple & Google have huge teams building their platforms & they will support them for as long as they exist.
- Native have a smaller footprint:
  - Typically, cross-platform apps are larger. (A simple “hello, world” app for Android might take up to 16MB, much of it being used by the associated libraries, content, Mono runtime, & Base Class Library assemblies.
- Volatile tools should not be depended on:
  - There are no guarantees that the cross-platform tool you are using will exist next month. These tools are fully dependant on the agenda of the companies funding them. They may not even be compatible with the next versions released.

What do we need?



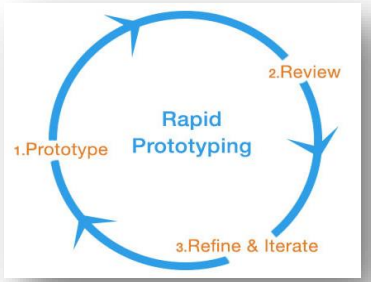
Exceptions to use Cross-platform tools?

Games:

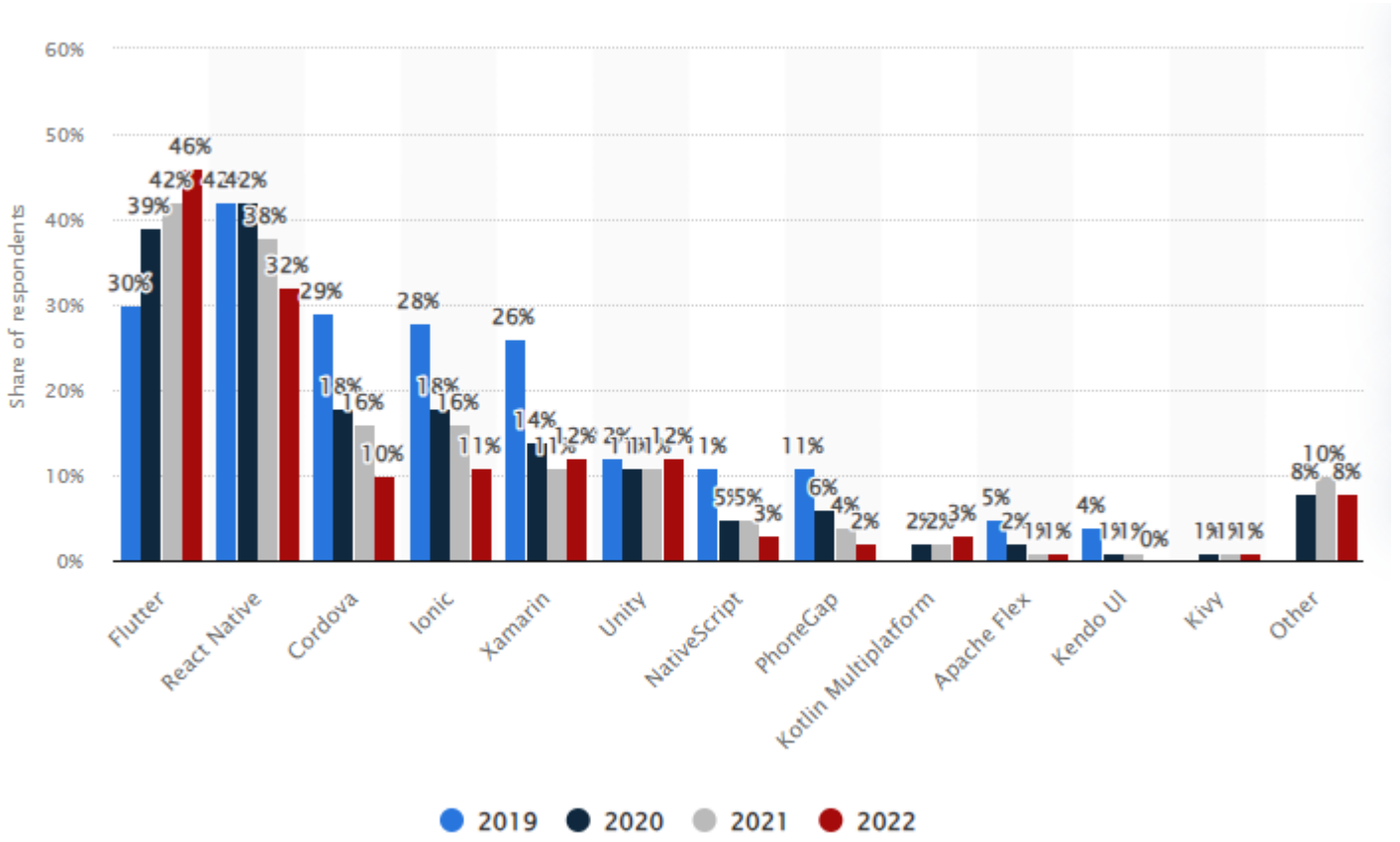
- Not dependent on native components. Usually use custom UI components designed to fit the game. Generally built on top of a graphics library, where performance is excellent, where visual logic makes most of the app. Can be shared across platforms.

Rapid prototyping:

- Good to quickly test an idea and deploy it to a group of testers, or where you are more interested in how your app interacts rather than whether it has good performance. Depending on a developer's skill, cross-platform can be a perfect fit for creating a prototype quickly.



Who's using what?



University of the West of England
BRISTOL

Top of the Bunch

A relatively new app framework developed by Google. Based on the Dart programming language, Flutter takes a different approach to implement cross-platform features by providing good performance & a native look. It combines the ease of development with native performance while maintaining visual consistency across platforms.

A JavaScript framework that offers localised apps for iOS & Android. Instead of aiming at the web browser, it targets the mobile platform. JavaScript allows you to create one app that runs on different platforms instead of creating separate apps for each system. You can build faster & cheaper applications as a result.

Employability

Best to focus on 1 platform to become more attractive to a mobile dev studio for employment.

You will always learn new skills and programming languages and tools are always evolving.

Nobody knows how to programme everything. But knowing how to build a mobile app in terms of **platform expectations** & **good mobile design** is highly desirable and what this module is about.

You can easily fall into a cross-platform development knowing more about each platforms development requirements which means you will make better apps!

HarmonyOS: a different Android?

HarmonyOS is Huawei's operating system designed for a wide range of devices, including smartphones, tablets, smart TVs, wearables, and IoT devices.

- 2012-2016: early development
- May 2019: US ban and HarmonyOS announcement
- August 2019: HarmonyOS 2.0 roadmap to expand to IoT devices
- June 2021: HarmonyOS 2.0 official release
  - Expansion to smartphones and tablets
  - Integration with Huawei devices to replace Android
  - Independent app ecosystem and development kits, but it is still compatible with Android
- July 2022: HarmonyOS 3.0 Official release
  - Super device, Cable-free Projection and better Privacy and Security
- August 2023: HarmonyOS 4.0 Official release
  - Not compatible with Android anymore

What new from another two?

- Cross-devices platform to run on a wide range of devices for a seamless experience.
- Distributed technology to allow devices to collaborate and share resources.
- Open source to be available to the public (Android is open to manufacturers only)
- App ecosystem for supporting other platforms
- Development language supports C, C++, Java, JavaScript, and Kotlin.

Does HarmonyOS have the future?

- Grown fast as firmly the third largest mobile OS.
- But, mainly focus on Chinese domestic market.

- HarmonyOS is not a copy of Android, nor is it a copy of iOS. Distribution is the major contribution it brings to the mobile Oss.