The Future of Cross-Platform is Native

Justin Mancinelli
justin@touchlab.co
@piannaf

TOUCHLAB

Efficient Developers

Efficient Developers

More Features

Efficient Developers

More Features

Fewer Bugs

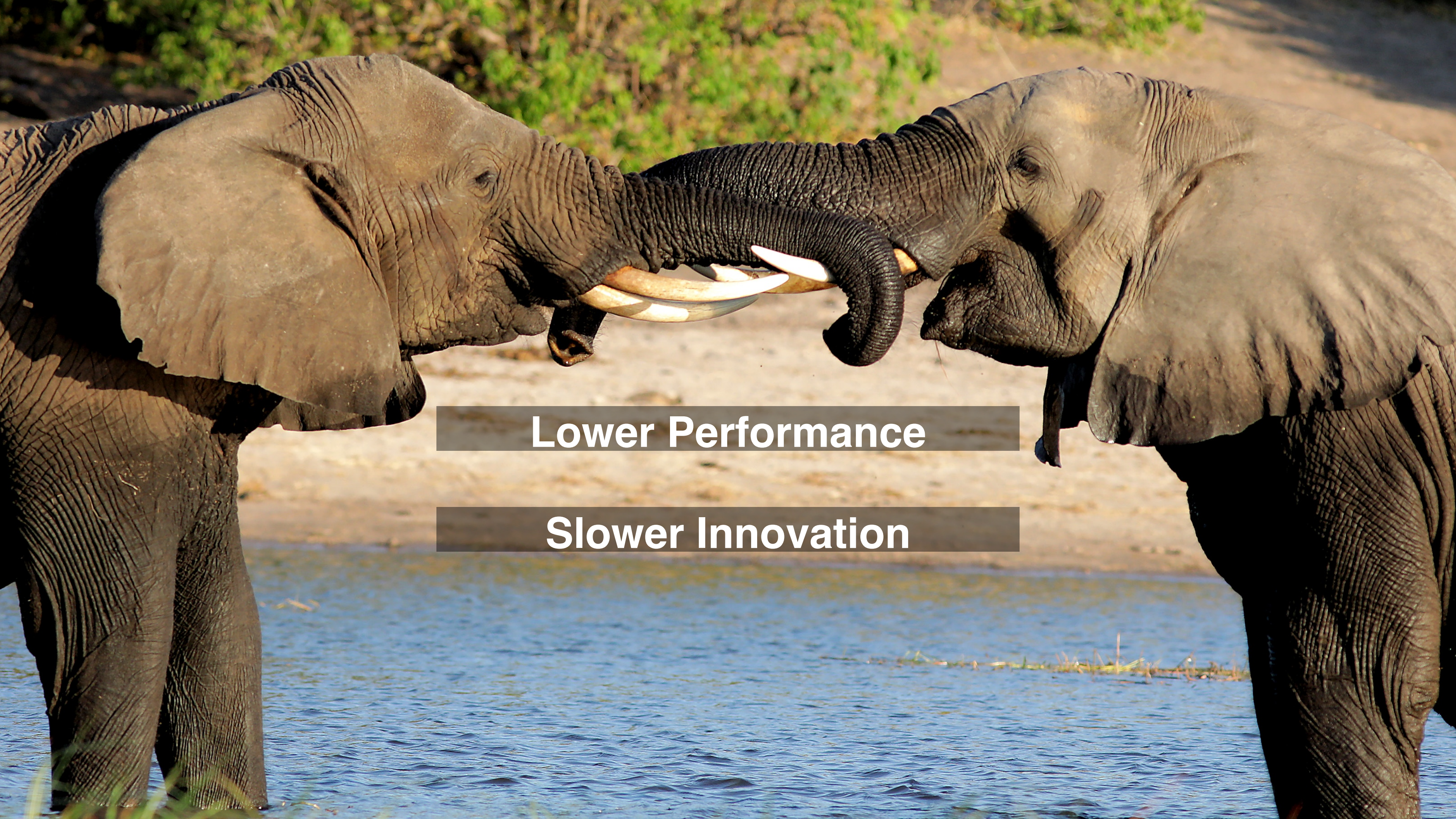
Efficient Developers

More Features

Fewer Bugs

Reach all the Users

Lower Performance



Lower Performance

Slower Innovation

Lower Performance

Slower Innovation

Poor UI

Lower Performance

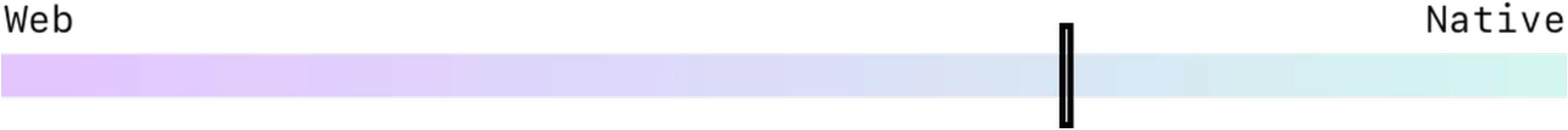
Slower Innovation

Poor UI

Vendor Lock-in

Cross-Platform?

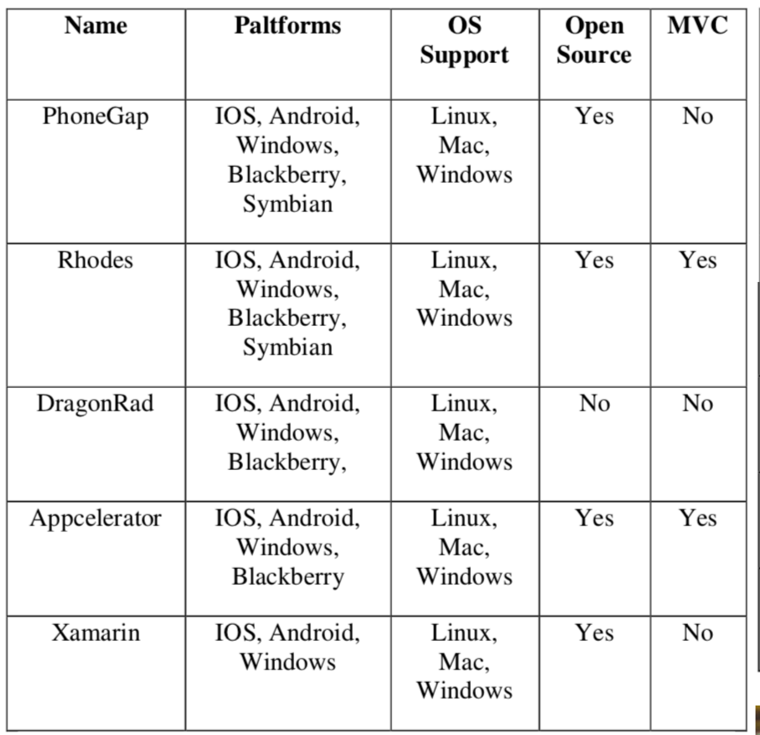
We Don’t Say That Around Here Anymore



Web

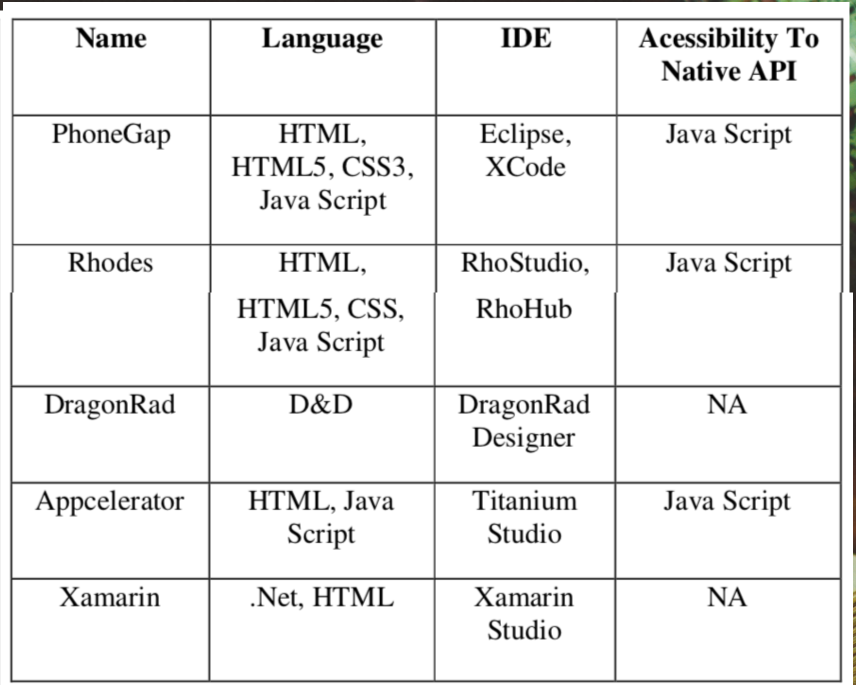
Native

Research



Name	Paltforms	OS Support	Open Source	MVC
PhoneGap	IOS, Android, Windows, Blackberry, Symbian	Linux, Mac, Windows	Yes	No
Rhodes	IOS, Android, Windows, Blackberry, Symbian	Linux, Mac, Windows	Yes	Yes
DragonRad	IOS, Android, Windows, Blackberry,	Linux, Mac, Windows	No	No
Appcelerator	IOS, Android, Windows, Blackberry	Linux, Mac, Windows	Yes	Yes
Xamarin	IOS, Android, Windows	Linux, Mac, Windows	Yes	No

Table 1 General Features



Name	Language	IDE	Acessibility To Native API
PhoneGap	HTML, HTML5, CSS3, Java Script	Eclipse, XCode	Java Script
Rhodes	HTML, HTML5, CSS, Java Script	RhoStudio, RhoHub	Java Script
DragonRad	D&D	DragonRad Designer	NA
Appcelerator	HTML, Java Script	Titanium Studio	Java Script
Xamarin	.Net, HTML	Xamarin Studio	NA

Table 2 Development Features

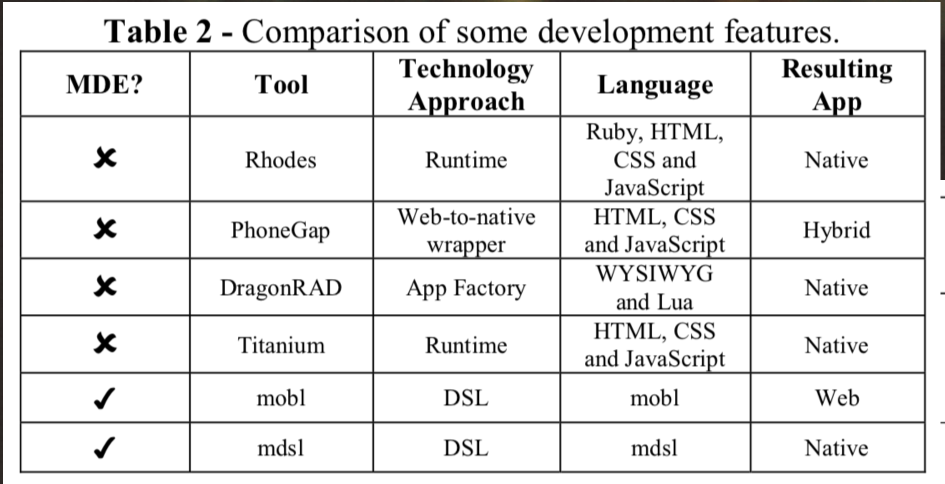
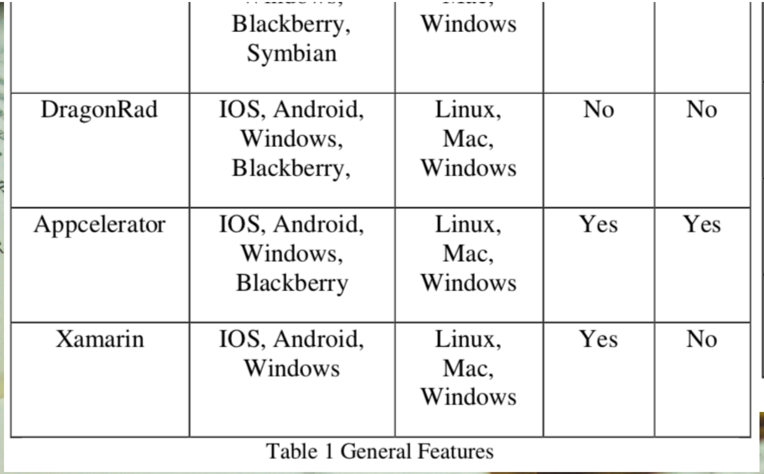


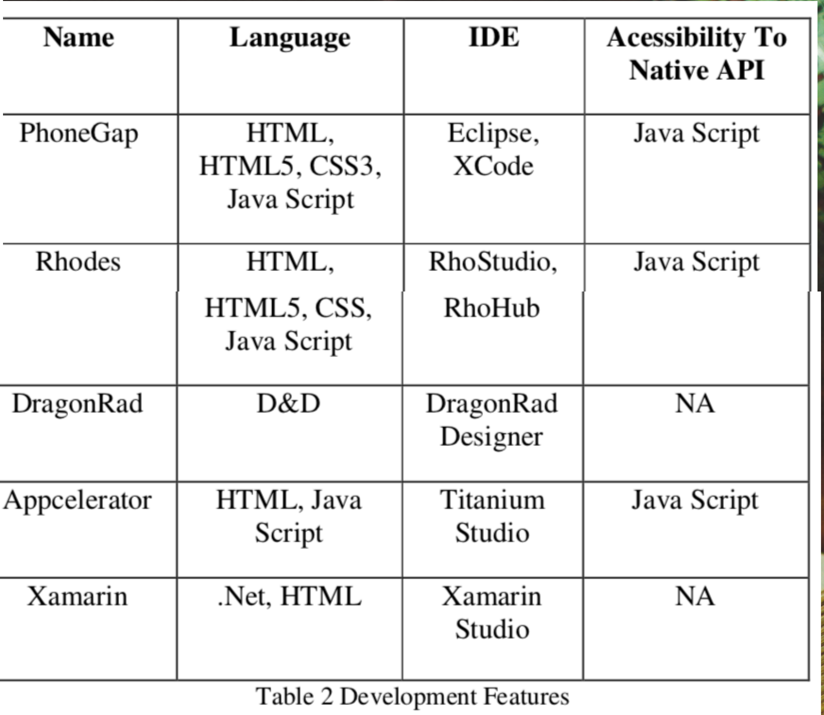
Table 2 - Comparison of some development features.

MDE?	Tool	Technology Approach	Language	Resulting App
✗	Rhodes	Runtime	Ruby, HTML, CSS and JavaScript	Native
✗	PhoneGap	Web-to-native wrapper	HTML, CSS and JavaScript	Hybrid
✗	DragonRAD	App Factory	WYSIWYG and Lua	Native
✗	Titanium	Runtime	HTML, CSS and JavaScript	Native
✓	mobl	DSL	mobl	Web
✓	mdsl	DSL	mdsl	Native



	Blackberry, Symbian	Windows		
DragonRad	IOS, Android, Windows, Blackberry,	Linux, Mac, Windows	No	No
Appcelerator	IOS, Android, Windows, Blackberry	Linux, Mac, Windows	Yes	Yes
Xamarin	IOS, Android, Windows	Linux, Mac, Windows	Yes	No

Table 1 General Features



Name	Language	IDE	Acessibility To Native API
PhoneGap	HTML, HTML5, CSS3, Java Script	Eclipse, XCode	Java Script
Rhodes	HTML, HTML5, CSS, Java Script	RhoStudio, RhoHub	Java Script
DragonRad	D&D	DragonRad Designer	NA
Appcelerator	HTML, Java Script	Titanium Studio	Java Script
Xamarin	.Net, HTML	Xamarin Studio	NA

Table 2 Development Features

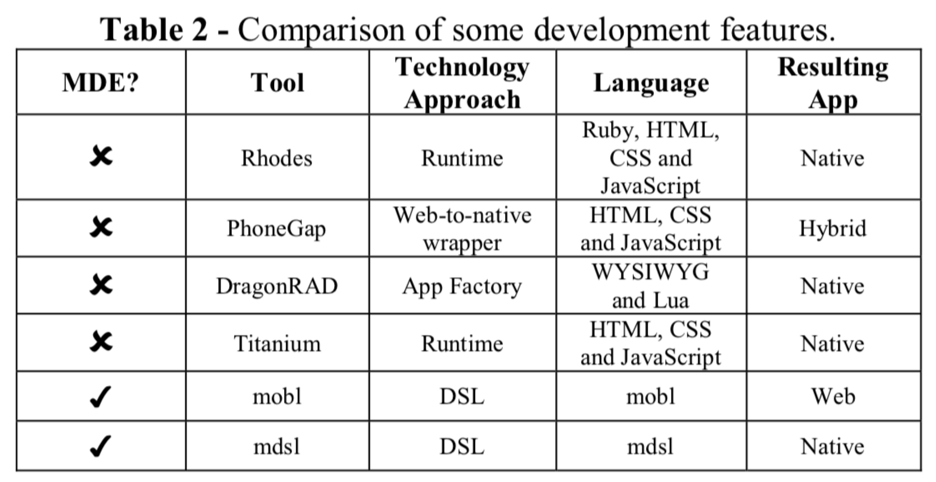
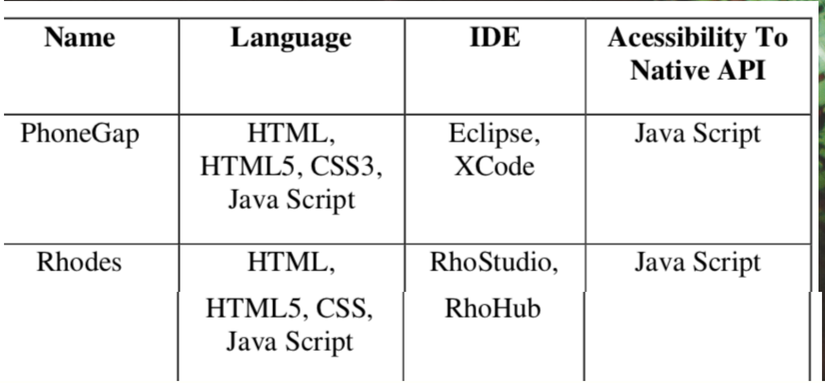


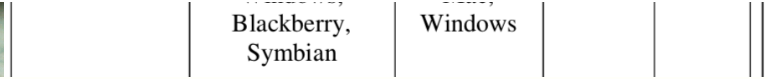
Table 2 - Comparison of some development features.

MDE?	Tool	Technology Approach	Language	Resulting App
✗	Rhodes	Runtime	Ruby, HTML, CSS and JavaScript	Native
✗	PhoneGap	Web-to-native wrapper	HTML, CSS and JavaScript	Hybrid
✗	DragonRAD	App Factory	WYSIWYG and Lua	Native
✗	Titanium	Runtime	HTML, CSS and JavaScript	Native
✓	mobl	DSL	mobl	Web
✓	mdsl	DSL	mdsl	Native



Name	Language	IDE	Acessibility To Native API
PhoneGap	HTML, HTML5, CSS3, Java Script	Eclipse, XCode	Java Script
Rhodes	HTML, HTML5, CSS, Java Script	RhoStudio, RhoHub	Java Script

Table 2 Development Features



	Blackberry, Symbian	Windows		

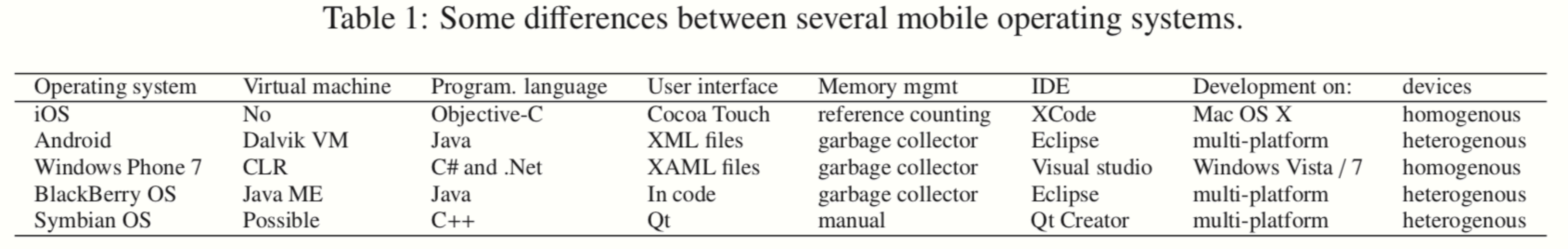
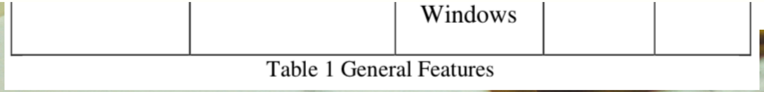


Table 1: Some differences between several mobile operating systems.

Operating system	Virtual machine	Program. language	User interface	Memory mgmt	IDE	Development on:	devices
iOS	No	Objective-C	Cocoa Touch	reference counting	XCode	Mac OS X	homogenous
Android	Dalvik VM	Java	XML files	garbage collector	Eclipse	multi-platform	heterogenous
Windows Phone 7	CLR	C# and .Net	XAML files	garbage collector	Visual studio	Windows Vista / 7	homogenous
BlackBerry OS	Java ME	Java	In code	garbage collector	Eclipse	multi-platform	heterogenous
Symbian OS	Possible	C++	Qt	manual	Qt Creator	multi-platform	heterogenous



		Windows		

Table 1 General Features

Table 2

MDE?
✗
✗
✗
✗
✓
✓

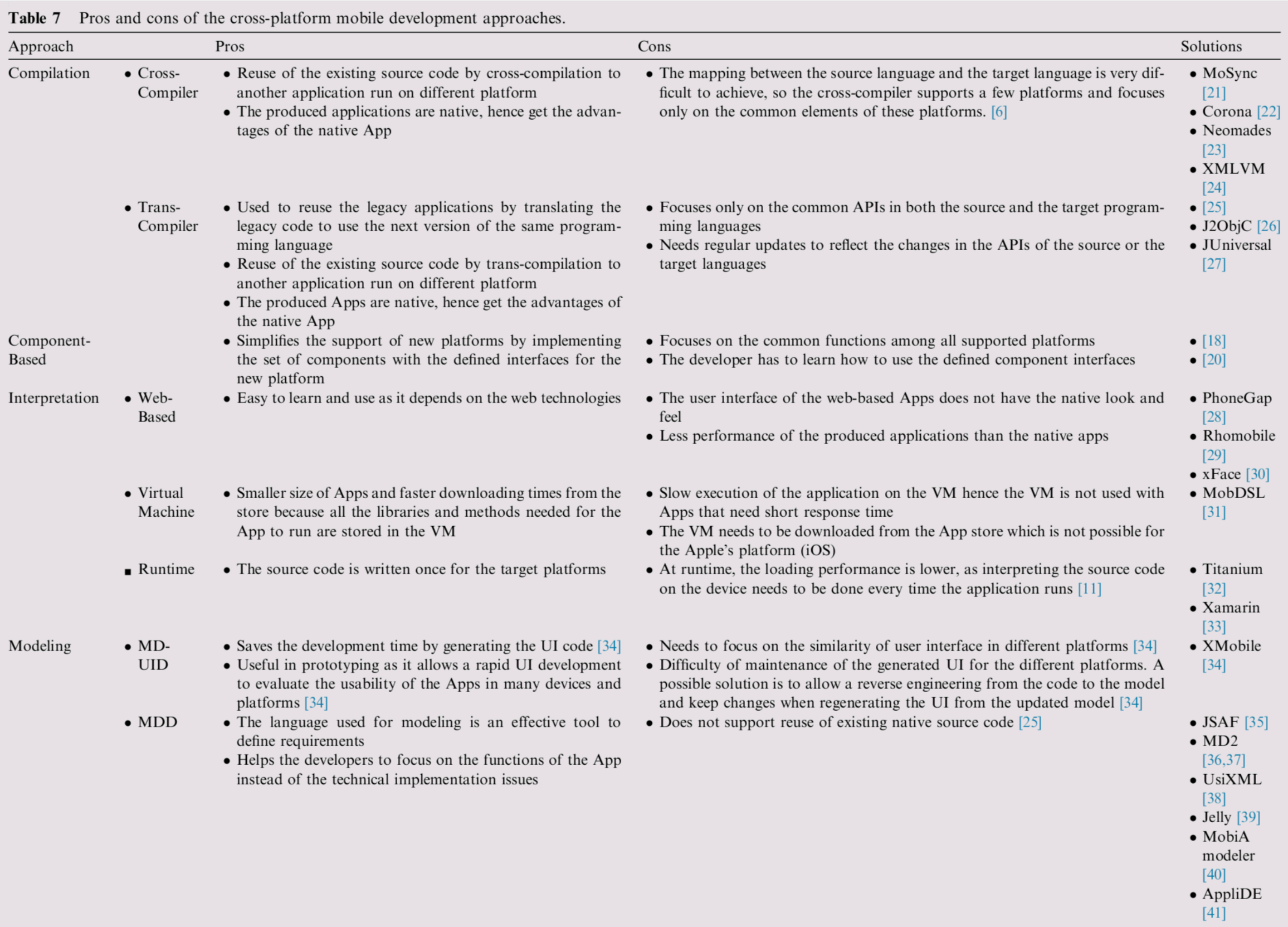
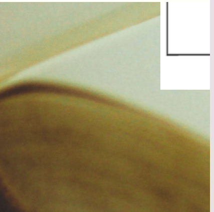


Table 7 Pros and cons of the cross-platform mobile development approaches.

Approach		Pros	Cons	Solutions
Compilation	• Cross-Compiler	• Reuse of the existing source code by cross-compilation to another application run on different platform • The produced applications are native, hence get the advantages of the native App	• The mapping between the source language and the target language is very difficult to achieve, so the cross-compiler supports a few platforms and focuses only on the common elements of these platforms. [6]	• MoSync [21] • Corona [22] • Neomades [23] • XMLVM [24]
	• Trans-Compiler	• Used to reuse the legacy applications by translating the legacy code to use the next version of the same programming language • Reuse of the existing source code by trans-compilation to another application run on different platform • The produced Apps are native, hence get the advantages of the native App	• Focuses only on the common APIs in both the source and the target programming languages • Needs regular updates to reflect the changes in the APIs of the source or the target languages	• [25] • J2ObjC [26] • JUniversal [27]
Component-Based		• Simplifies the support of new platforms by implementing the set of components with the defined interfaces for the new platform	• Focuses on the common functions among all supported platforms • The developer has to learn how to use the defined component interfaces	• [18] • [20]
Interpretation	• Web-Based	• Easy to learn and use as it depends on the web technologies	• The user interface of the web-based Apps does not have the native look and feel • Less performance of the produced applications than the native apps	• PhoneGap [28] • Rhomobile [29] • xFace [30]
	• Virtual Machine	• Smaller size of Apps and faster downloading times from the store because all the libraries and methods needed for the App to run are stored in the VM	• Slow execution of the application on the VM hence the VM is not used with Apps that need short response time • The VM needs to be downloaded from the App store which is not possible for the Apple's platform (iOS)	• MobDSL [31]
	▪ Runtime	• The source code is written once for the target platforms	• At runtime, the loading performance is lower, as interpreting the source code on the device needs to be done every time the application runs [11]	• Titanium [32] • Xamarin [33]
Modeling	• MD-UID	• Saves the development time by generating the UI code [34] • Useful in prototyping as it allows a rapid UI development to evaluate the usability of the Apps in many devices and platforms [34]	• Needs to focus on the similarity of user interface in different platforms [34] • Difficulty of maintenance of the generated UI for the different platforms. A possible solution is to allow a reverse engineering from the code to the model and keep changes when regenerating the UI from the updated model [34]	• XMobile [34]
	• MDD	• The language used for modeling is an effective tool to define requirements • Helps the developers to focus on the functions of the App instead of the technical implementation issues	• Does not support reuse of existing native source code [25]	• JSAF [35] • MD2 [36,37] • UsiXML [38] • Jelly [39] • MobiA modeler [40] • AppliDE [41]

Operating syst	devices
iOS	homogenous
Android	heterogenous
Windows Phon	homogenous
BlackBerry OS	heterogenous
Symbian OS	heterogenous



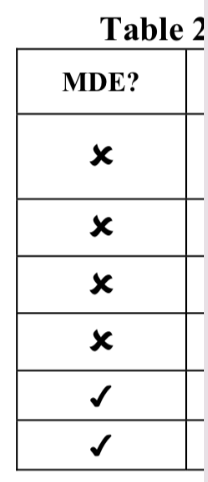


Table 2

MDE?
✗
✗
✗
✗
✓
✓

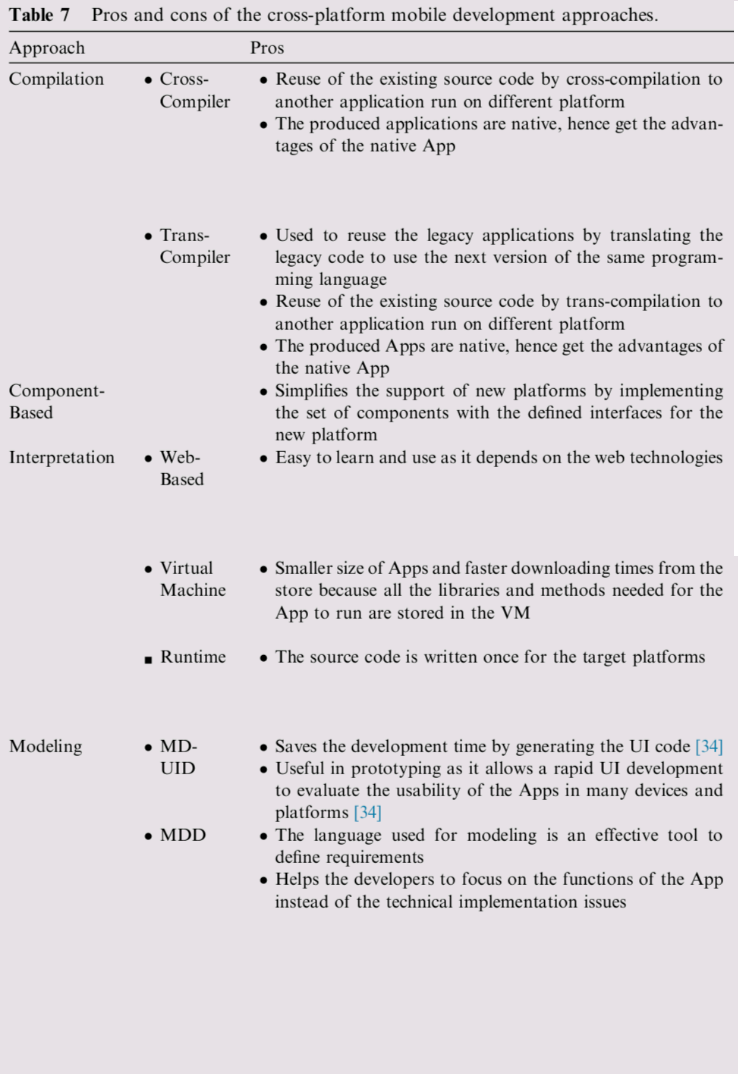


Table 7 Pros and cons of the cross-platform mobile development approaches.

Approach		Pros		
Compilation	• Cross-Compiler	• Reuse of the existing source code by cross-compilation to another application run on different platform • The produced applications are native, hence get the advantages of the native App		
	• Trans-Compiler	• Used to reuse the legacy applications by translating the legacy code to use the next version of the same programming language • Reuse of the existing source code by trans-compilation to another application run on different platform • The produced Apps are native, hence get the advantages of the native App		
Component-Based		• Simplifies the support of new platforms by implementing the set of components with the defined interfaces for the new platform		
Interpretation	• Web-Based	• Easy to learn and use as it depends on the web technologies		
	• Virtual Machine	• Smaller size of Apps and faster downloading times from the store because all the libraries and methods needed for the App to run are stored in the VM	• Slow execution of the application on the VM hence the VM is not used with Apps that need short response time • The VM needs to be downloaded from the App store which is not possible for the Apple's platform (iOS)	• MobDSL [31]
	▪ Runtime	• The source code is written once for the target platforms	• At runtime, the loading performance is lower, as interpreting the source code on the device needs to be done every time the application runs [11]	• Titanium [32] • Xamarin [33]
Modeling	• MD-UID	• Saves the development time by generating the UI code [34] • Useful in prototyping as it allows a rapid UI development to evaluate the usability of the Apps in many devices and platforms [34]	• Needs to focus on the similarity of user interface in different platforms [34] • Difficulty of maintenance of the generated UI for the different platforms. A possible solution is to allow a reverse engineering from the code to the model and keep changes when regenerating the UI from the updated model [34]	• XMobile [34]
	• MDD	• The language used for modeling is an effective tool to define requirements • Helps the developers to focus on the functions of the App instead of the technical implementation issues	• Does not support reuse of existing native source code [25]	• JSAF [35] • MD2 [36,37] • UsiXML [38] • Jelly [39] • MobiA modeler [40] • AppliDE [41]

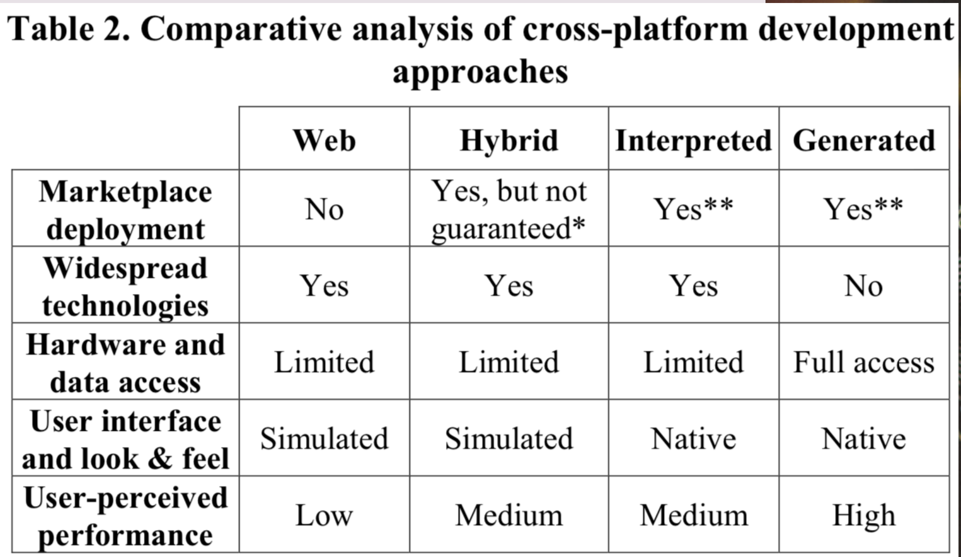
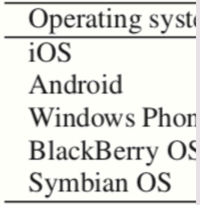


Table 2. Comparative analysis of cross-platform development approaches

	Web	Hybrid	Interpreted	Generated
Marketplace deployment	No	Yes, but not guaranteed*	Yes**	Yes**
Widespread technologies	Yes	Yes	Yes	No
Hardware and data access	Limited	Limited	Limited	Full access
User interface and look & feel	Simulated	Simulated	Native	Native
User-perceived performance	Low	Medium	Medium	High



Operating syst
iOS
Android
Windows Phon
BlackBerry OS
Symbian OS

devices
homogenous
heterogenous
homogenous
heterogenous
heterogenous

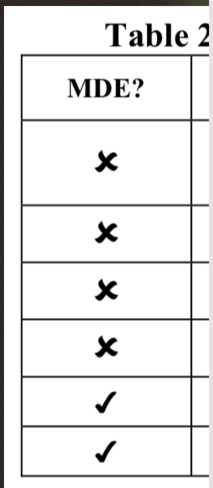
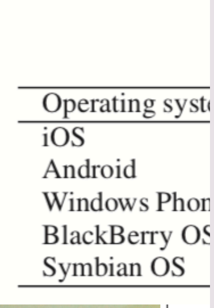


Table 2

MDE?
✗
✗
✗
✗
✓
✓



Operating syst
iOS
Android
Windows Phon
BlackBerry OS
Symbian OS

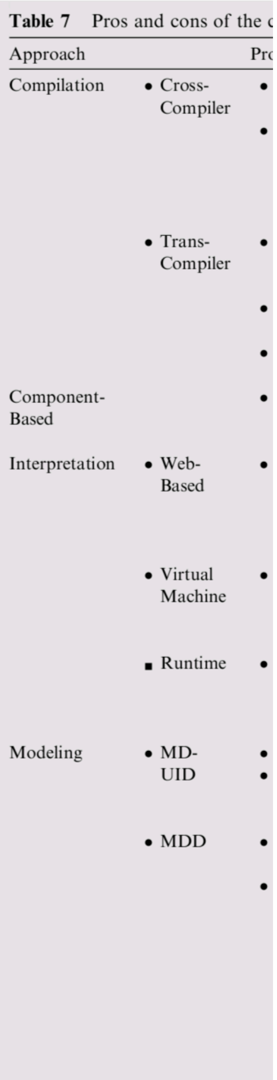


Table 7 Pros and cons of the c

Approach		Pro
Compilation	• Cross-Compiler	• •
	• Trans-Compiler	• • •
Component-Based		•
Interpretation	• Web-Based	•
	• Virtual Machine	•
	▪ Runtime	•
Modeling	• MD-UID	• •
	• MDD	• •

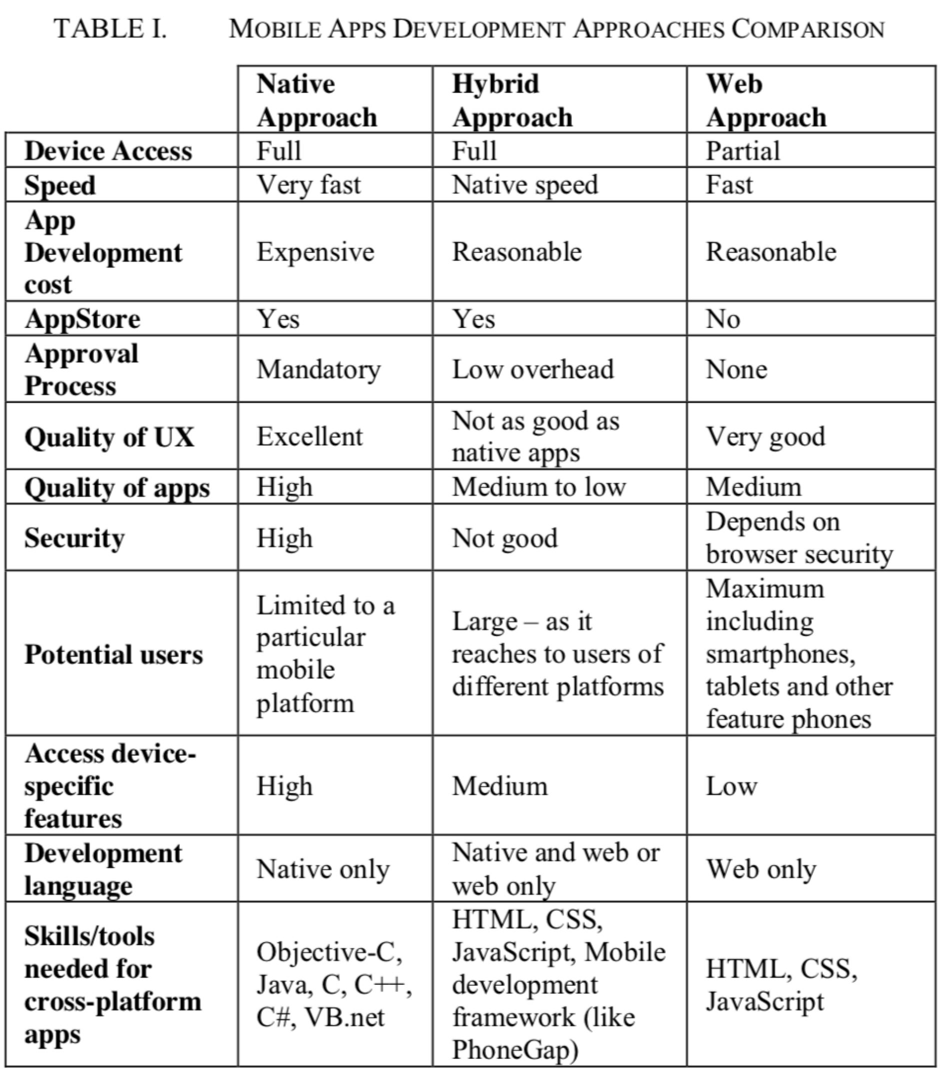
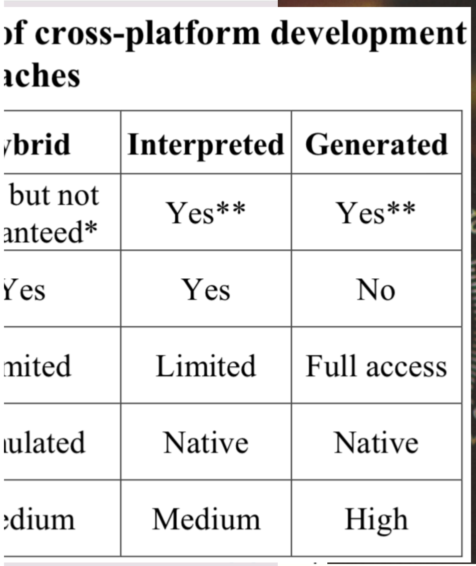


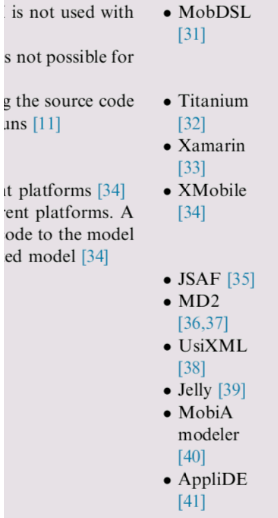
TABLE I. Mobile Apps Development Approaches Comparison

	Native Approach	Hybrid Approach	Web Approach
Device Access	Full	Full	Partial
Speed	Very fast	Native speed	Fast
App Development cost	Expensive	Reasonable	Reasonable
AppStore	Yes	Yes	No
Approval Process	Mandatory	Low overhead	None
Quality of UX	Excellent	Not as good as native apps	Very good
Quality of apps	High	Medium to low	Medium
Security	High	Not good	Depends on browser security
Potential users	Limited to a particular mobile platform	Large – as it reaches to users of different platforms	Maximum including smartphones, tablets and other feature phones
Access device-specific features	High	Medium	Low
Development language	Native only	Native and web or web only	Web only
Skills/tools needed for cross-platform apps	Objective-C, Java, C, C++, C#, VB.net	HTML, CSS, JavaScript, Mobile development framework (like PhoneGap)	HTML, CSS, JavaScript

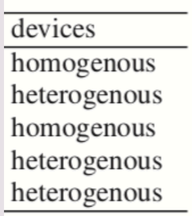


of cross-platform development aches

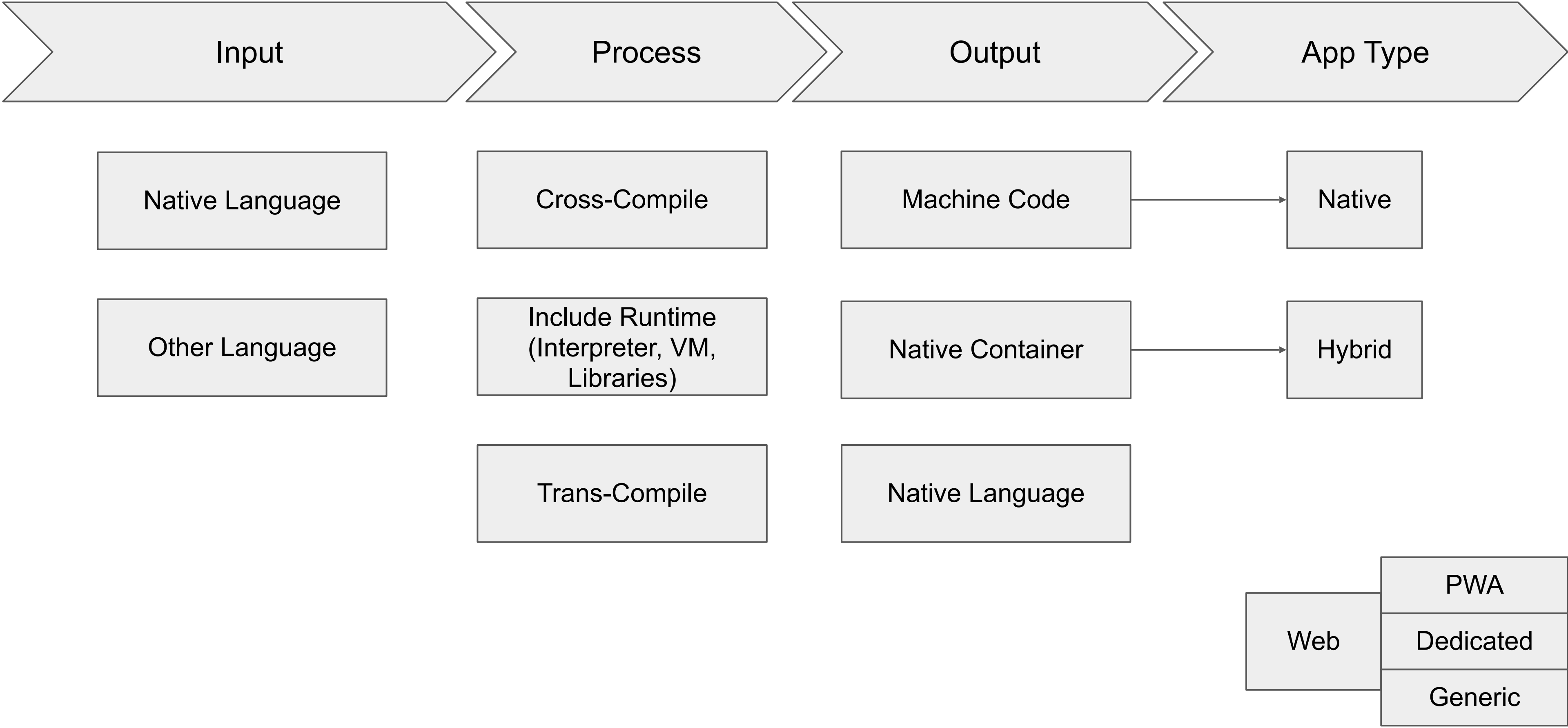
ybrid	Interpreted	Generated
but not anteed*	Yes**	Yes**
Yes	Yes	No
mited	Limited	Full access
ulated	Native	Native
edium	Medium	High



is not used with	• MobDSL [31]
s not possible for	
g the source code ns [11]	• Titanium [32] • Xamarin [33]
t platforms [34] ent platforms. A ode to the model ed model [34]	• XMobile [34]
	• JSAF [35] • MD2 [36,37] • UsiXML [38] • Jelly [39] • MobiA modeler [40] • AppliDE [41]



devices
homogenous
heterogenous
homogenous
heterogenous
heterogenous



Input

Process

Output

App Type

Native Language

Other Language

Cross-Compile

Include Runtime (Interpreter, VM, Libraries)

Trans-Compile

Machine Code

Native Container

Native Language

Native

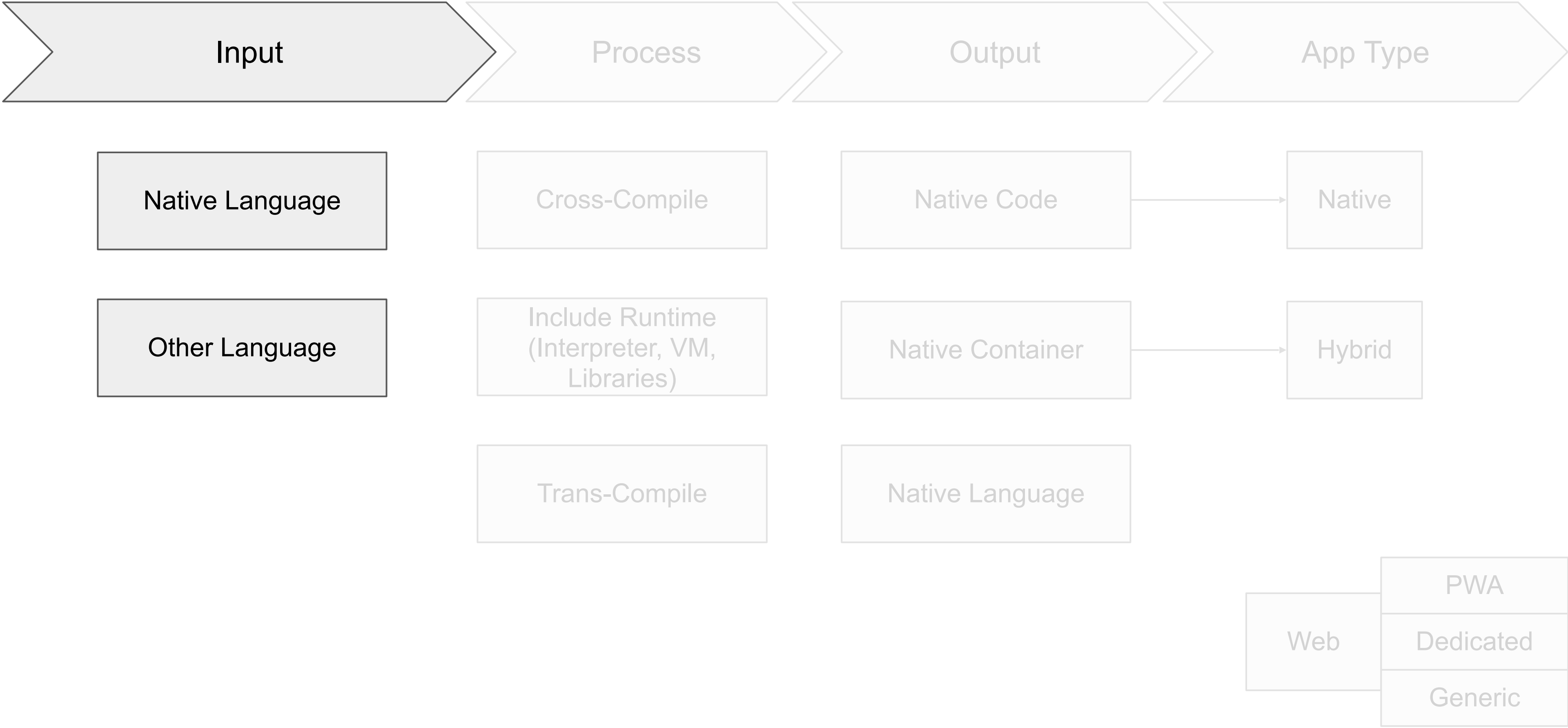
Hybrid

Web

PWA

Dedicated

Generic



Input

Native Language

Other Language

Process

Cross-Compile

Include Runtime (Interpreter, VM, Libraries)

Trans-Compile

Output

Native Code

Native Container

Native Language

App Type

Native

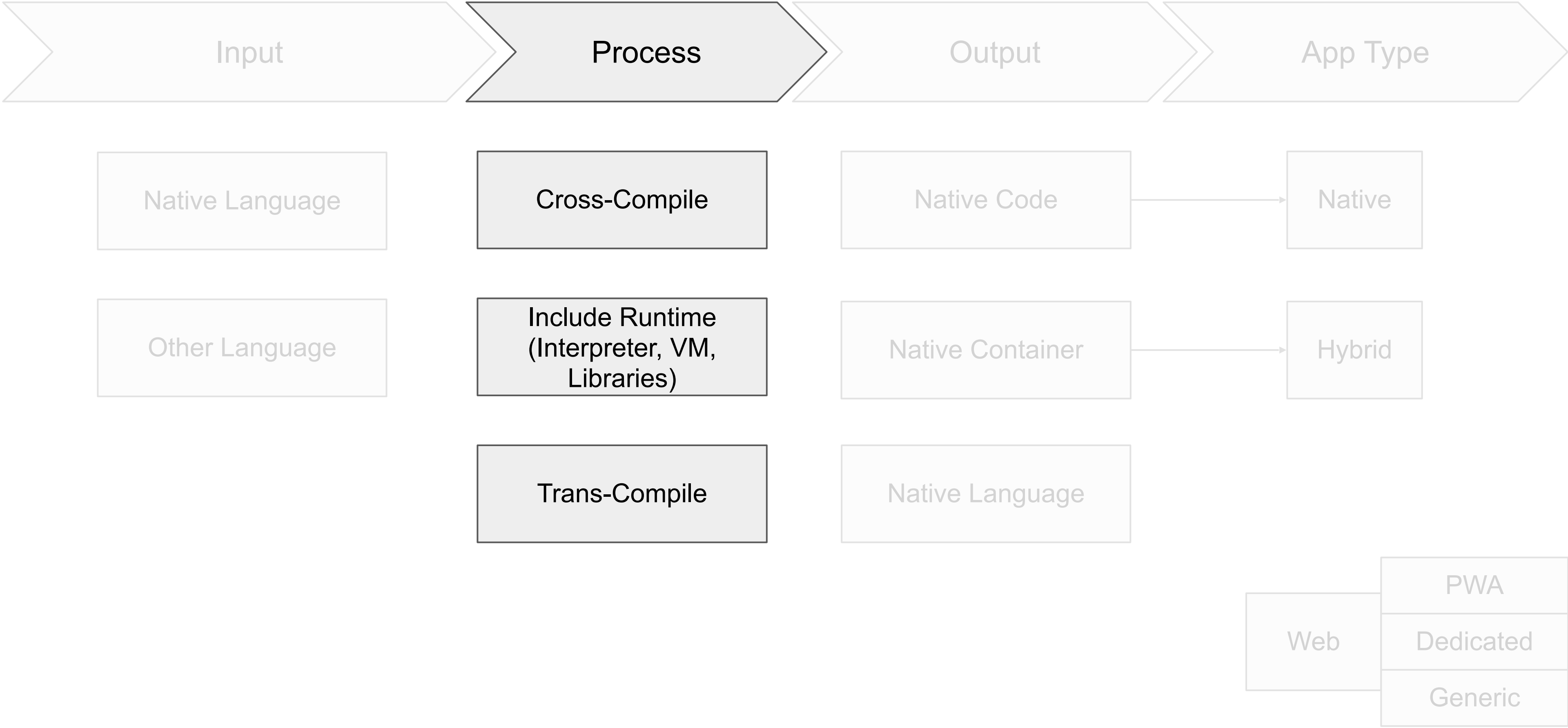
Hybrid

Web

PWA

Dedicated

Generic



Input

Process

Output

App Type

Native Language

Other Language

Cross-Compile

Include Runtime (Interpreter, VM, Libraries)

Trans-Compile

Native Code

Native Container

Native Language

Native

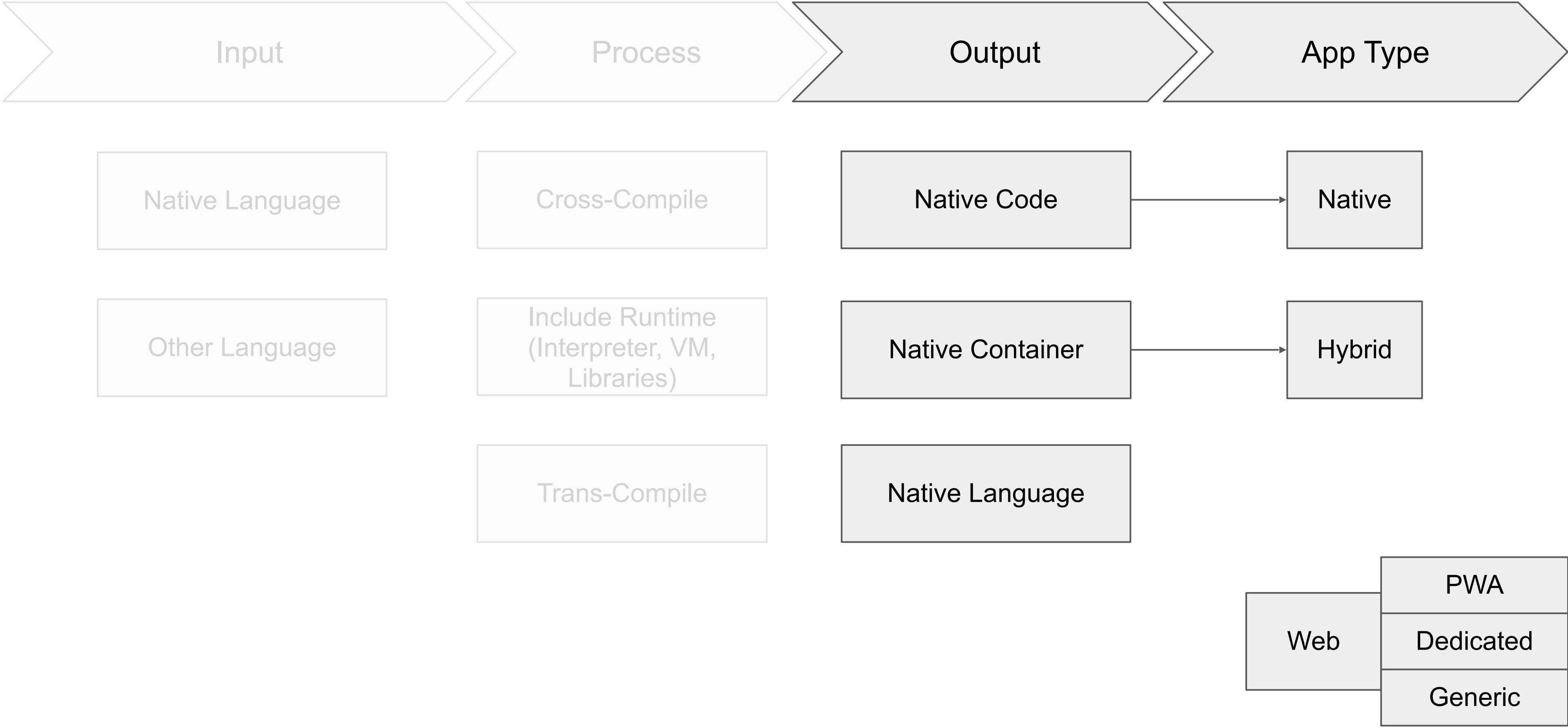
Hybrid

Web

PWA

Dedicated

Generic



Input

Process

Output

App Type

Native Language

Cross-Compile

Native Code

Native

Other Language

Include Runtime (Interpreter, VM, Libraries)

Native Container

Hybrid

Trans-Compile

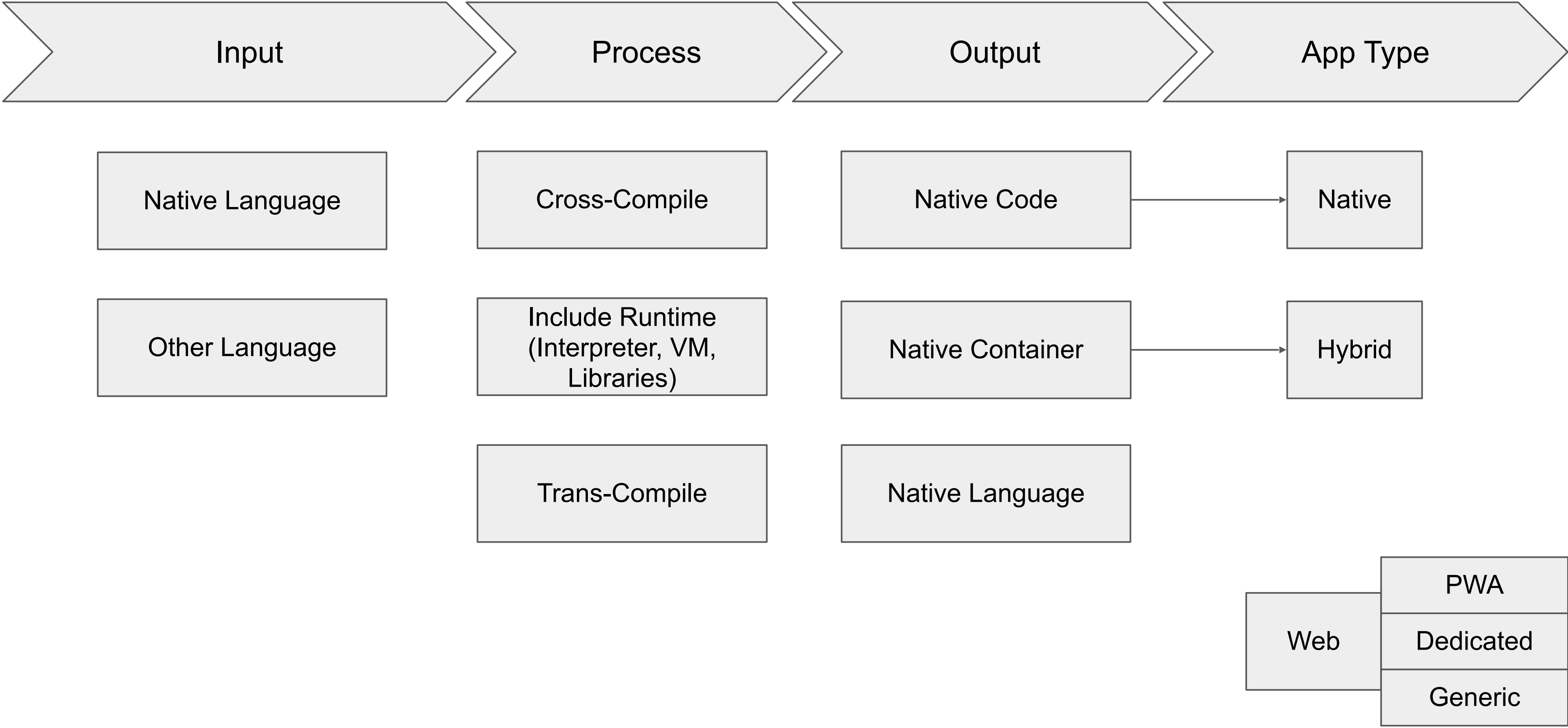
Native Language

Web

PWA

Dedicated

Generic



Input

Native Language

Other Language

Process

Cross-Compile

Include Runtime (Interpreter, VM, Libraries)

Trans-Compile

Output

Native Code

Native Container

Native Language

App Type

Native

Hybrid

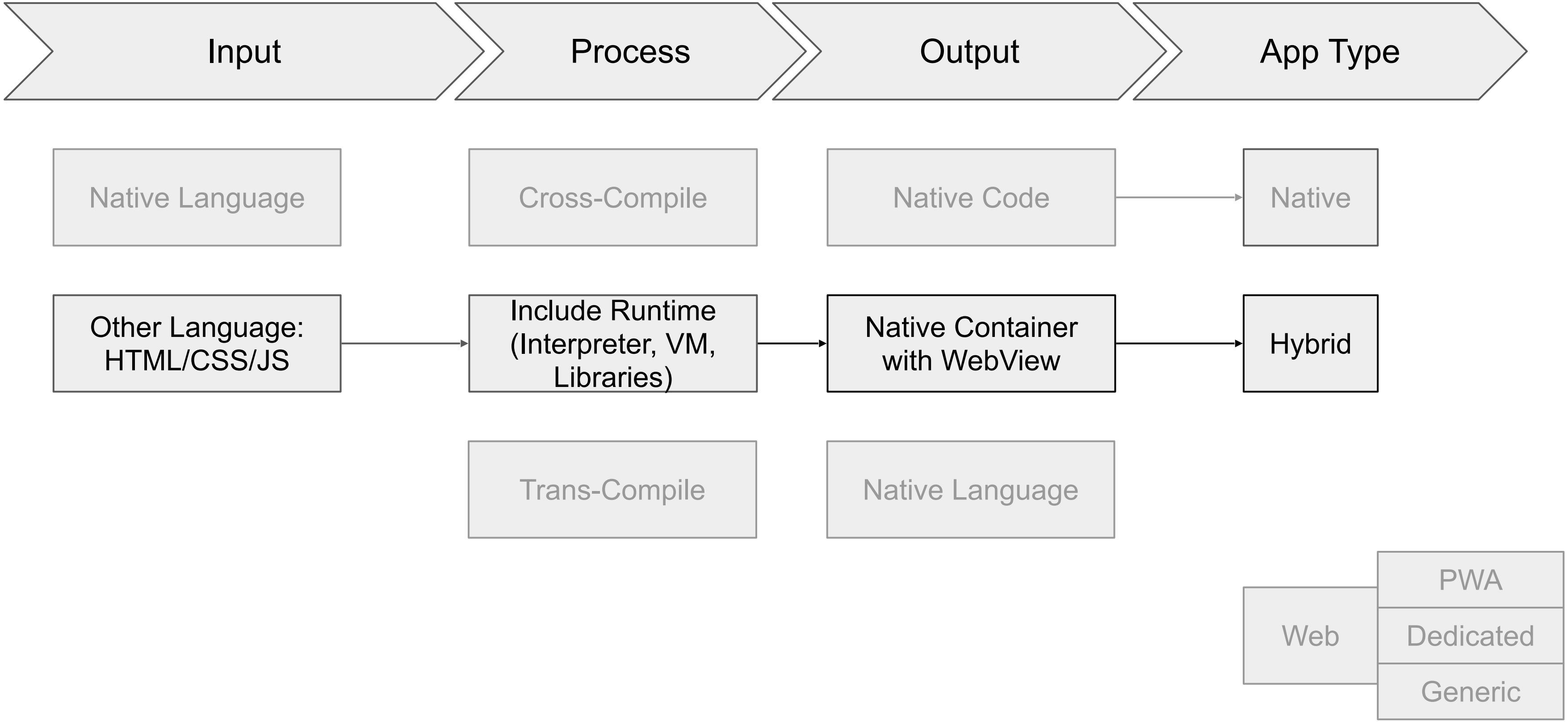
Web

PWA

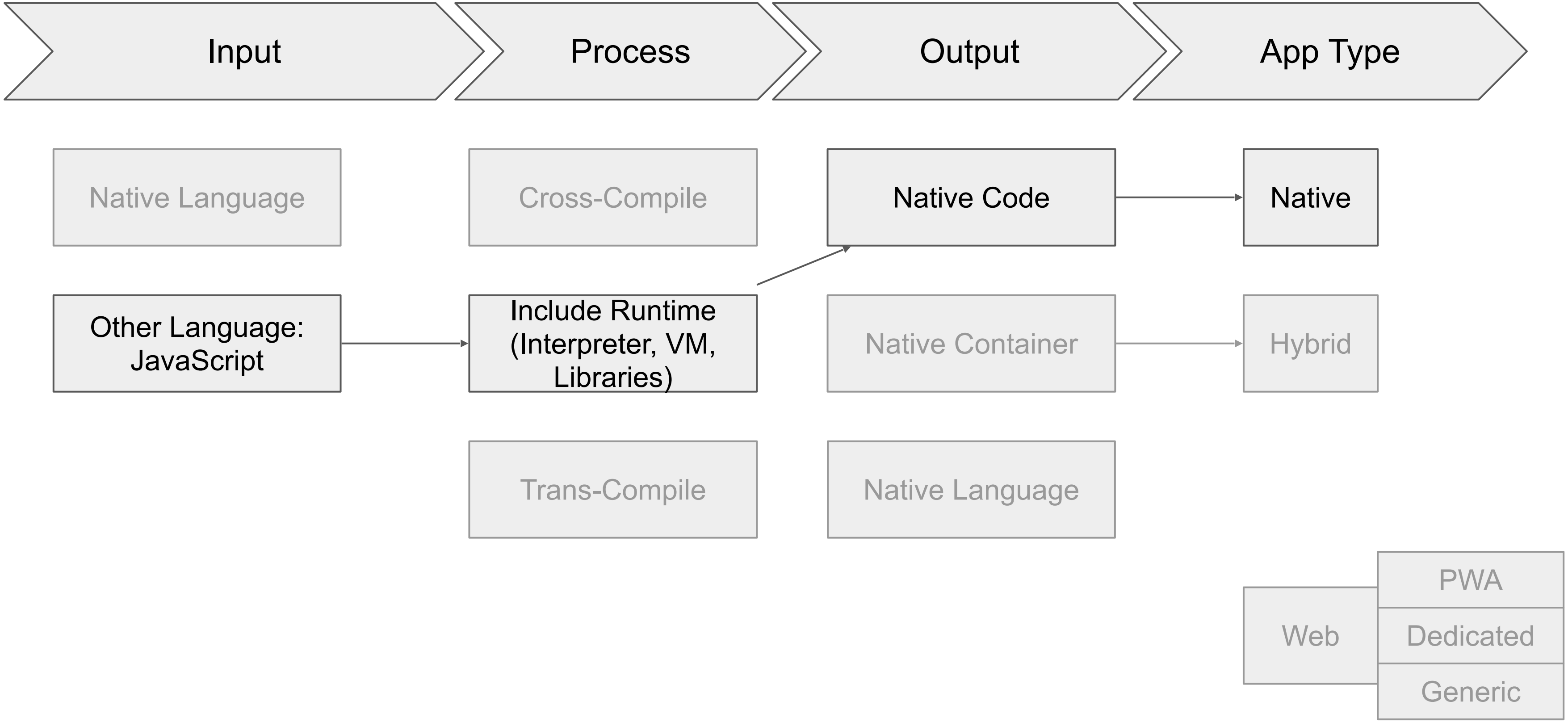
Dedicated

Generic

Ionic


Input
Process
Output
App Type
Native Language
Cross-Compile
Native Code
Native
Other Language: HTML/CSS/JS
Include Runtime (Interpreter, VM, Libraries)
Native Container with WebView
Hybrid
Trans-Compile
Native Language
Web
PWA
Dedicated
Generic

React Native



Input

Process

Output

App Type

Native Language

Cross-Compile

Native Code

Native

Other Language: JavaScript

Include Runtime (Interpreter, VM, Libraries)

Native Container

Hybrid

Trans-Compile

Native Language

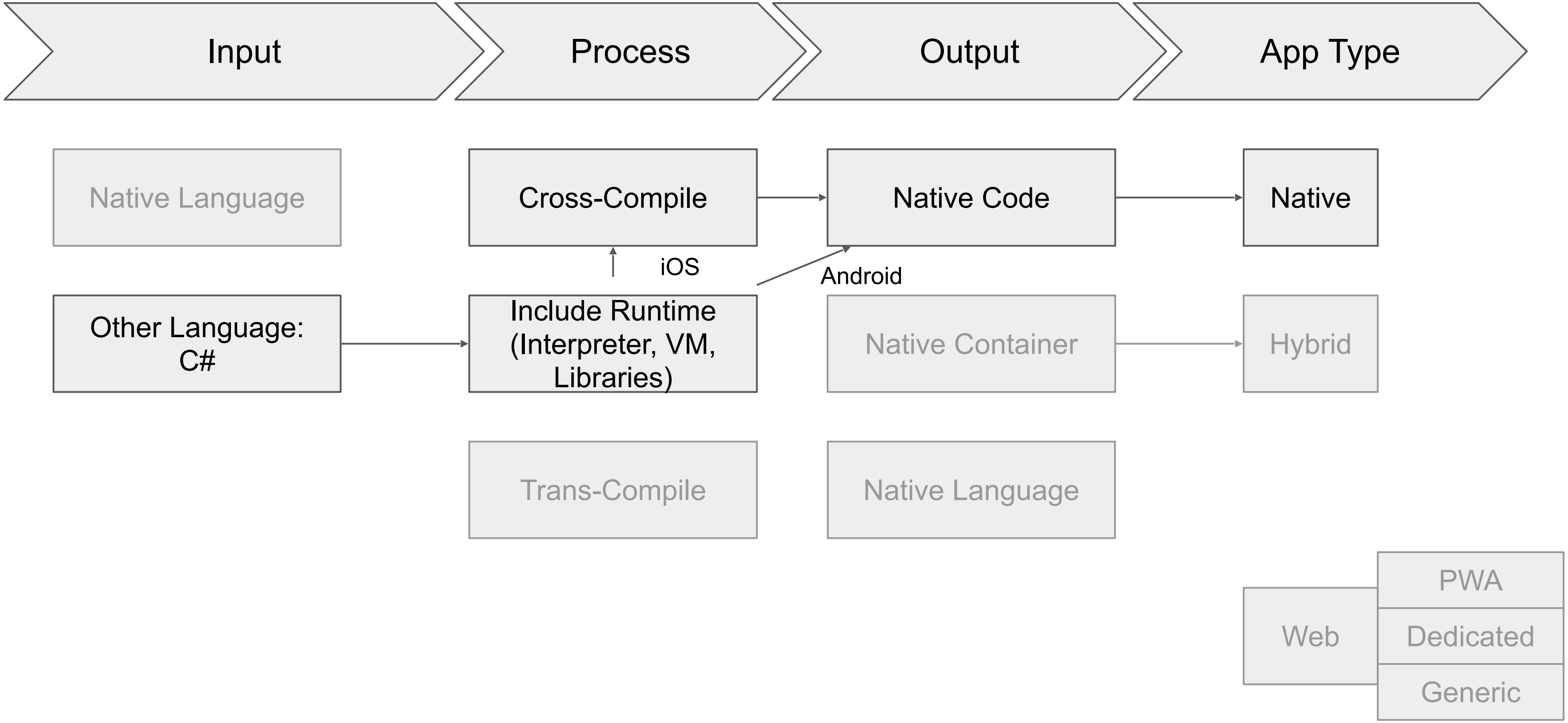
Web

PWA

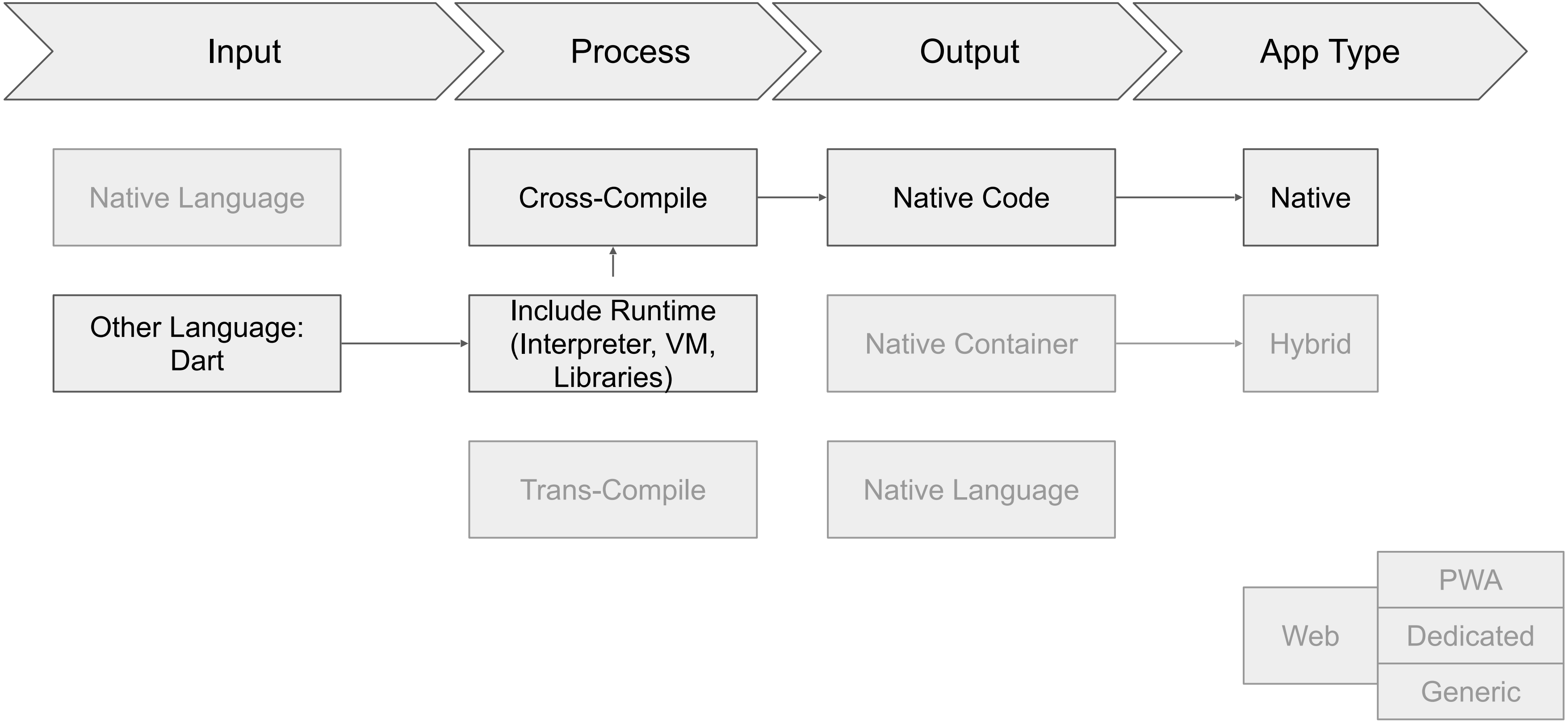
Dedicated

Generic

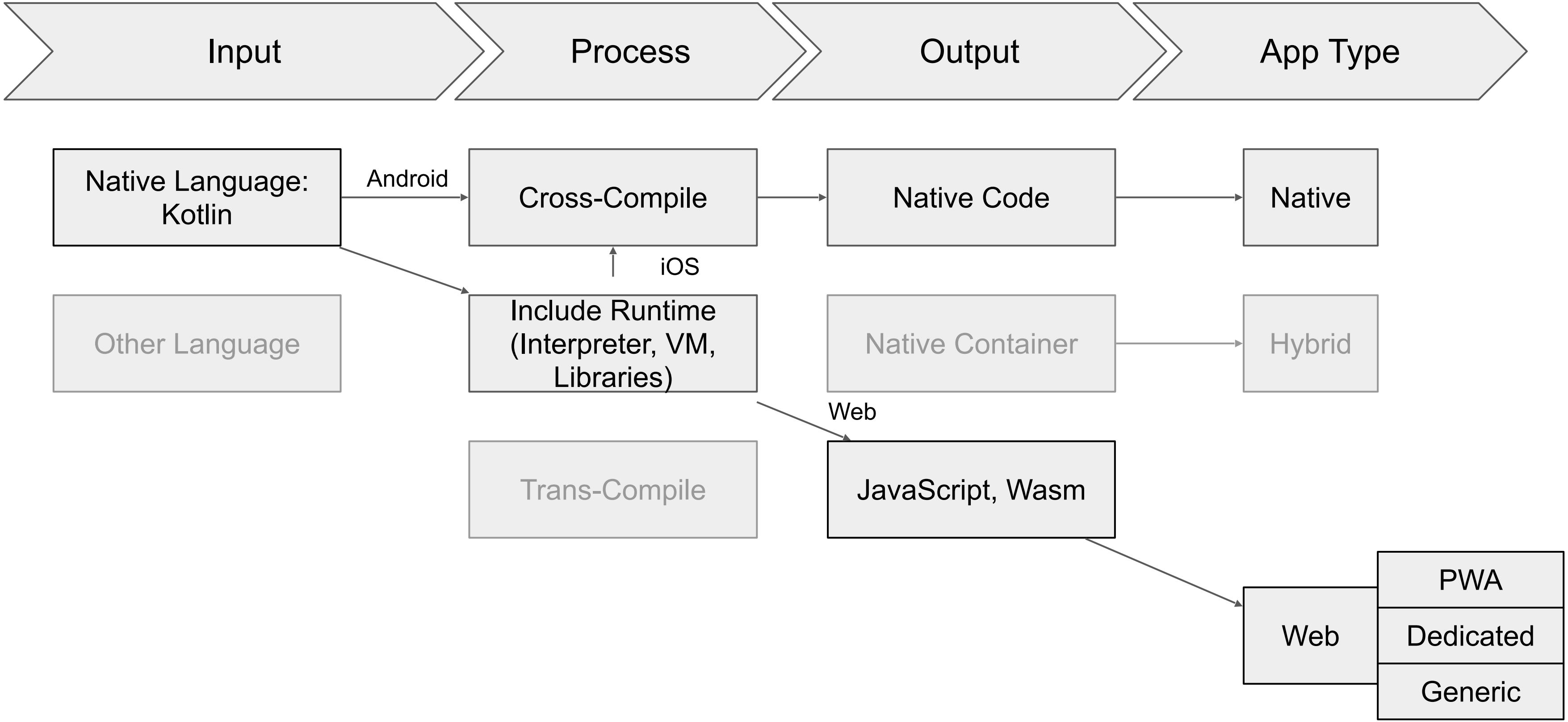
Xamarin

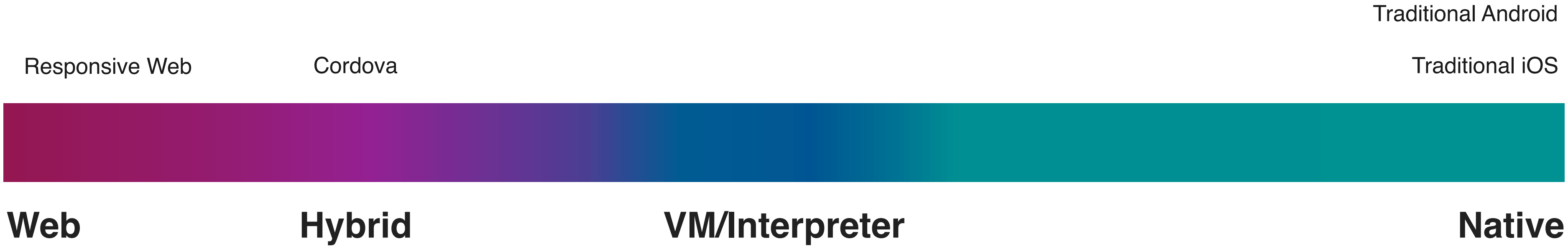

Input
Process
Output
App Type
Native Language
Cross-Compile
Native Code
Native
iOS
Android
Other Language: C#
Include Runtime (Interpreter, VM, Libraries)
Native Container
Hybrid
Trans-Compile
Native Language
Web
PWA
Dedicated
Generic

Flutter


Input
Process
Output
App Type
Native Language
Cross-Compile
Native Code
Native
Other Language: Dart
Include Runtime (Interpreter, VM, Libraries)
Native Container
Hybrid
Trans-Compile
Native Language
PWA
Web
Dedicated
Generic

Kotlin Multiplatform





Traditional Android

Responsive Web

Cordova

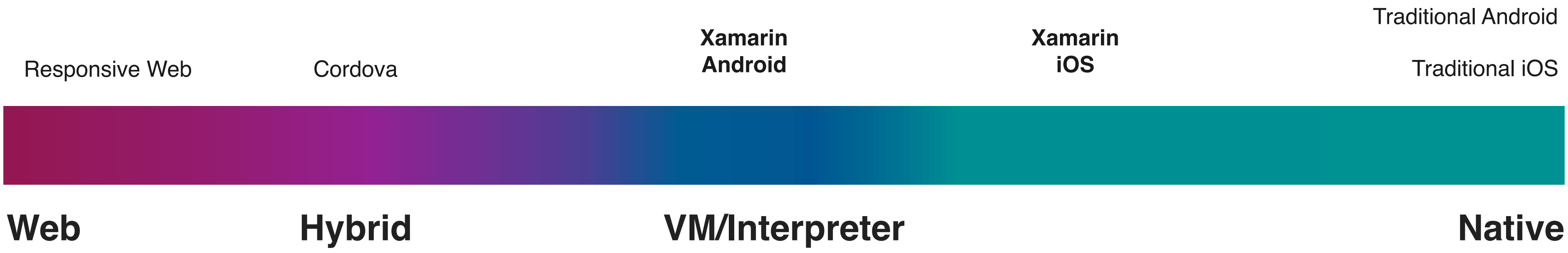
Traditional iOS

Web

Hybrid

VM/Interpreter

Native



Traditional Android

Xamarin
Android

Xamarin
iOS

Responsive Web

Cordova

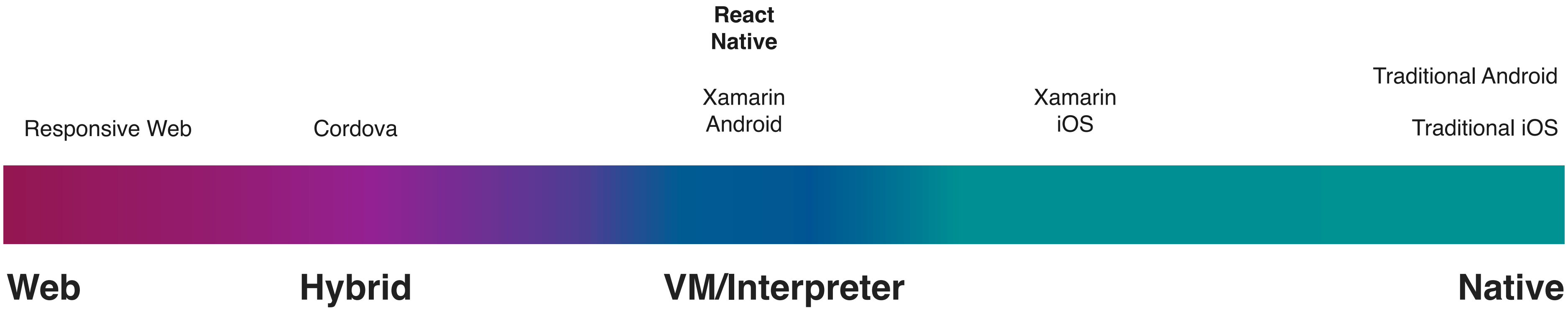
Traditional iOS

Web

Hybrid

VM/Interpreter

Native



React Native

Traditional Android

Xamarin Android

Xamarin iOS

Responsive Web

Cordova

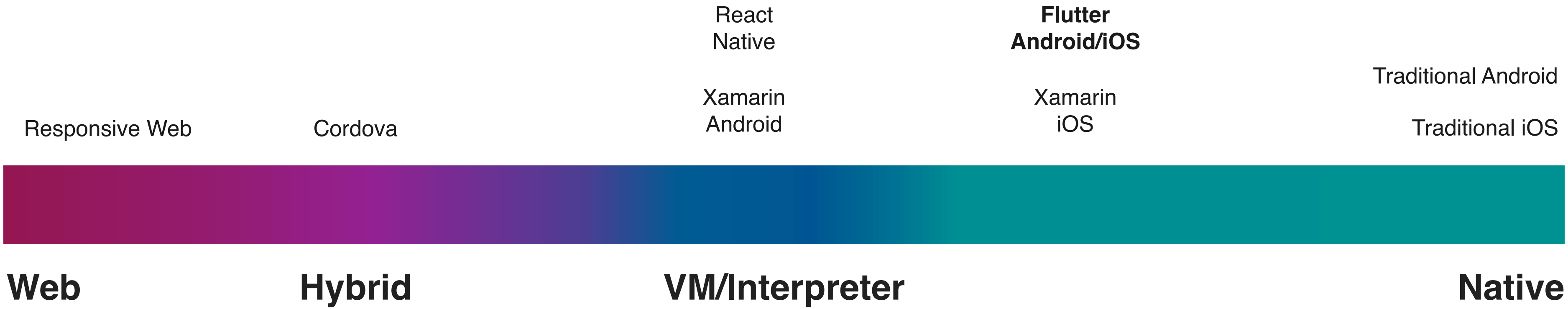
Traditional iOS

Web

Hybrid

VM/Interpreter

Native



Responsive Web

Cordova

React Native

Xamarin Android

Flutter Android/iOS

Xamarin iOS

Traditional Android

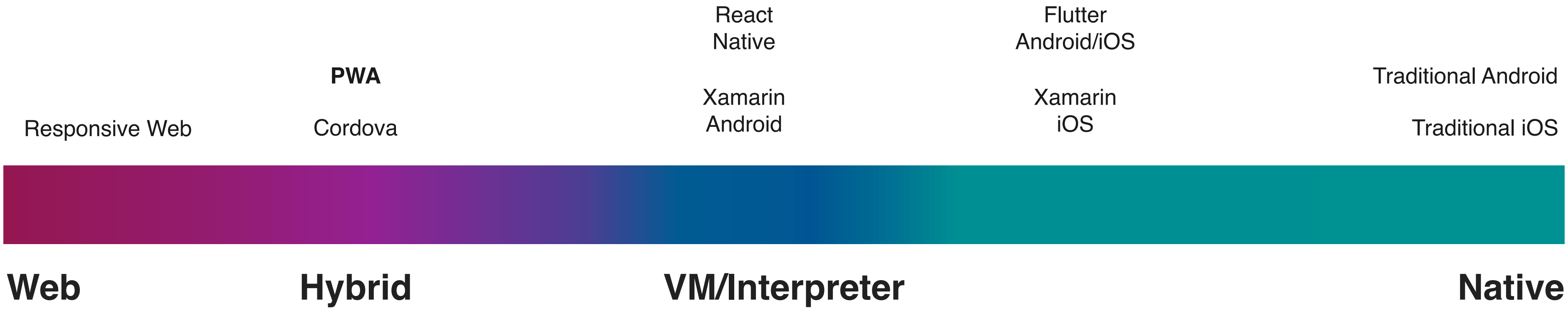
Traditional iOS

Web

Hybrid

VM/Interpreter

Native



Responsive Web

PWA

Cordova

React Native

Xamarin Android

Flutter Android/iOS

Xamarin iOS

Traditional Android

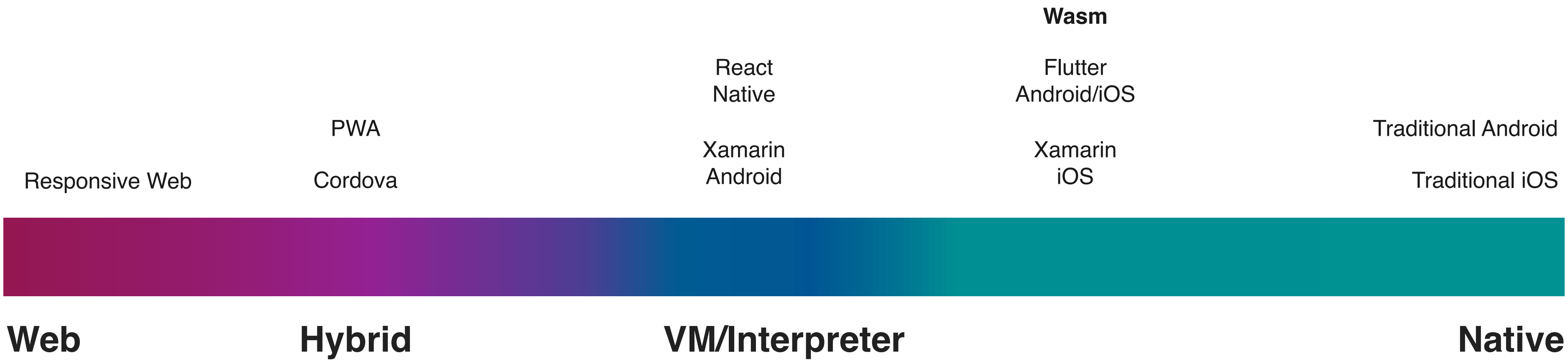
Traditional iOS

Web

Hybrid

VM/Interpreter

Native



Responsive Web

Web

PWA

Cordova

Hybrid

React Native

Xamarin Android

VM/Interpreter

Wasm

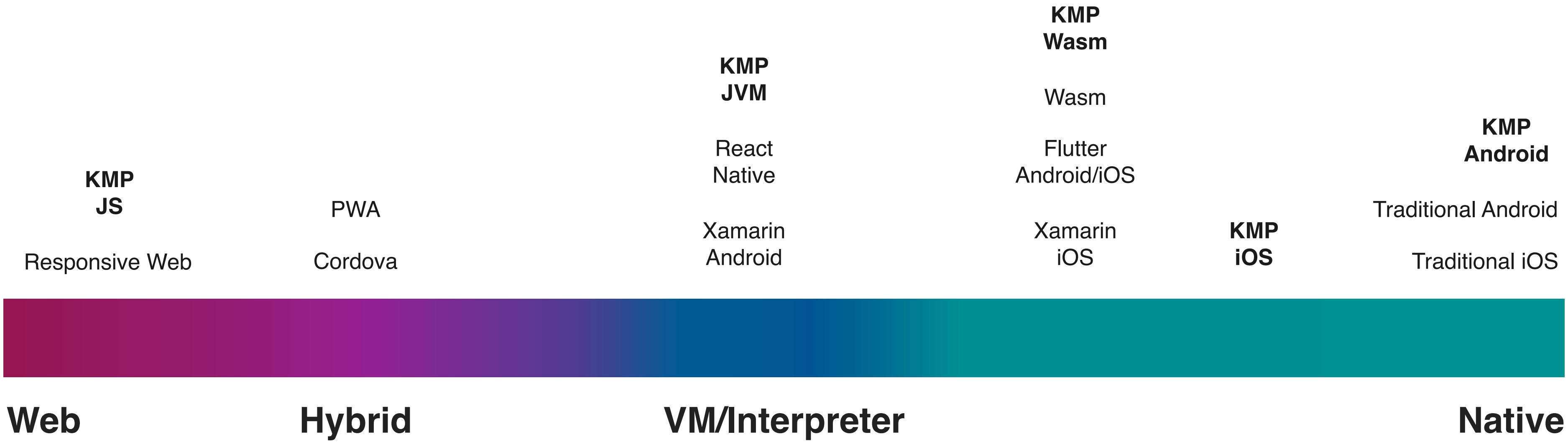
Flutter Android/iOS

Xamarin iOS

Traditional Android

Traditional iOS

Native



KMP
JS

Responsive Web

PWA

Cordova

KMP
JVM

React
Native

Xamarin
Android

KMP
Wasm

Wasm

Flutter
Android/iOS

Xamarin
iOS

KMP
iOS

KMP
Android

Traditional Android

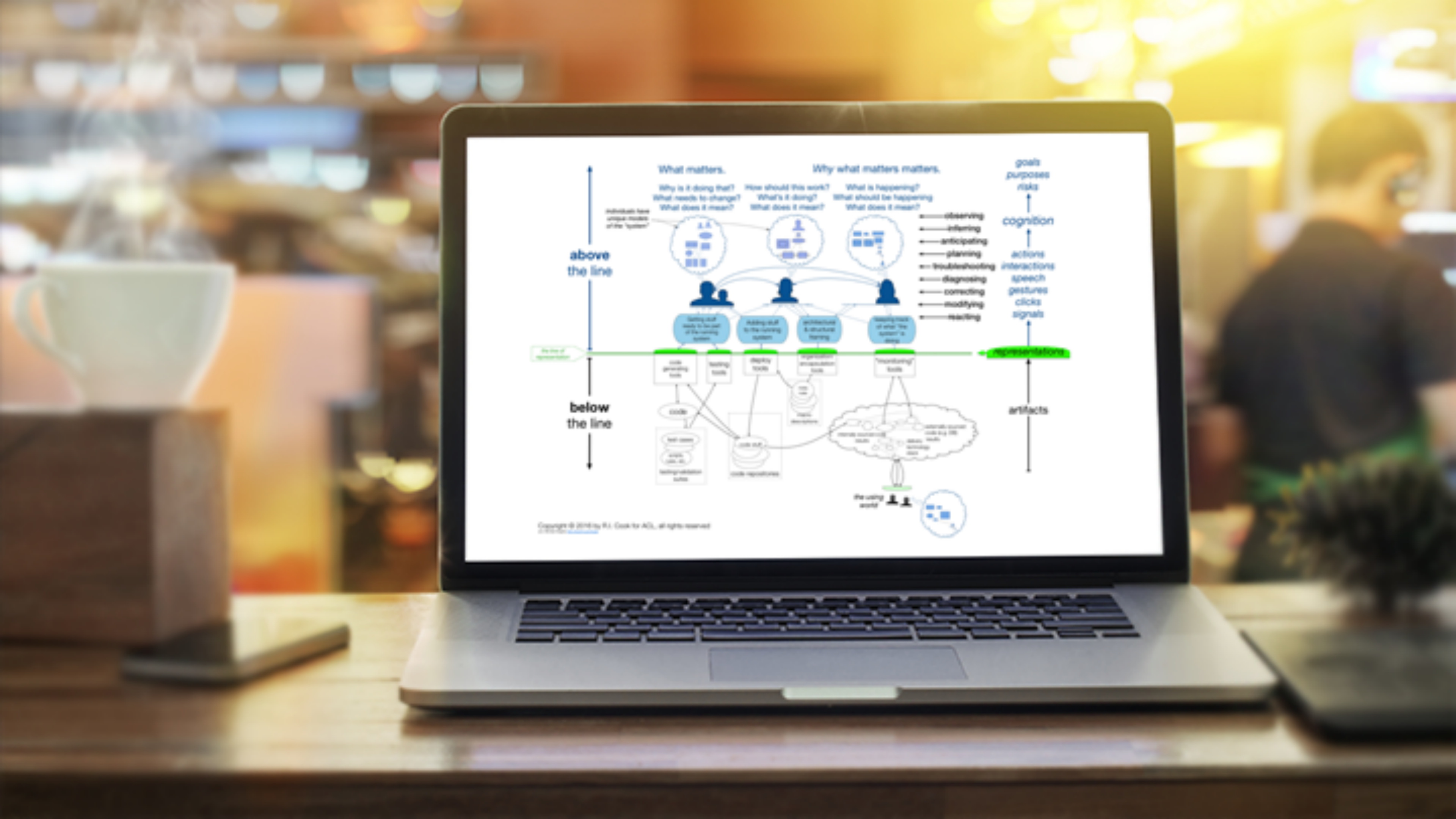
Traditional iOS

Web

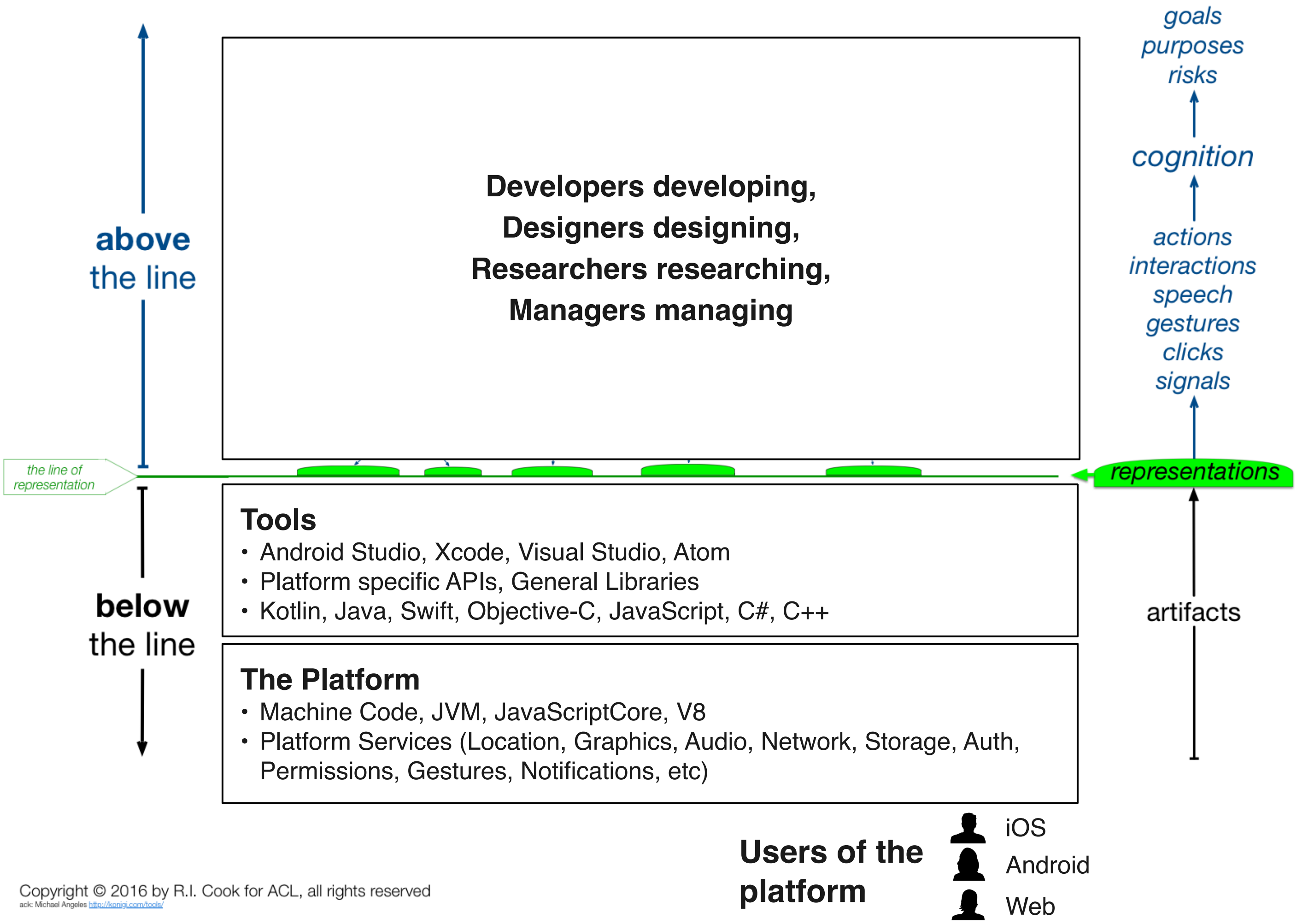
Hybrid

VM/Interpreter

Native



Copyright © 2016 by R.I. Cook for ACL, all rights reserved



above the line

Developers developing,
Designers designing,
Researchers researching,
Managers managing

goals
purposes
risks

cognition

actions
interactions
speech
gestures
clicks
signals

the line of representation

representations

below the line

Tools

- Android Studio, Xcode, Visual Studio, Atom
- Platform specific APIs, General Libraries
- Kotlin, Java, Swift, Objective-C, JavaScript, C#, C++

The Platform

- Machine Code, JVM, JavaScriptCore, V8
- Platform Services (Location, Graphics, Audio, Network, Storage, Auth, Permissions, Gestures, Notifications, etc)

artifacts

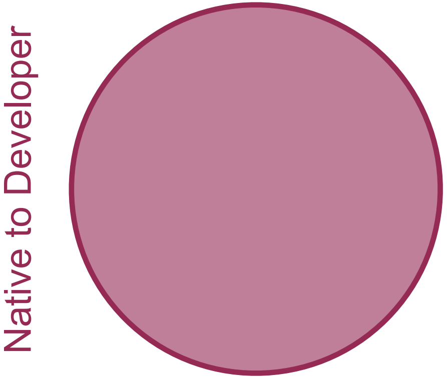
Users of the platform

iOS
Android
Web

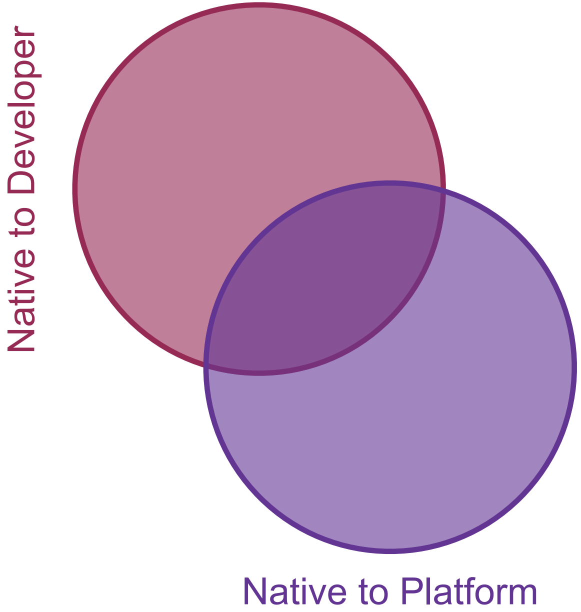
ack: Michael Angeles http://konigi.com/tools/

Changing Perspectives:

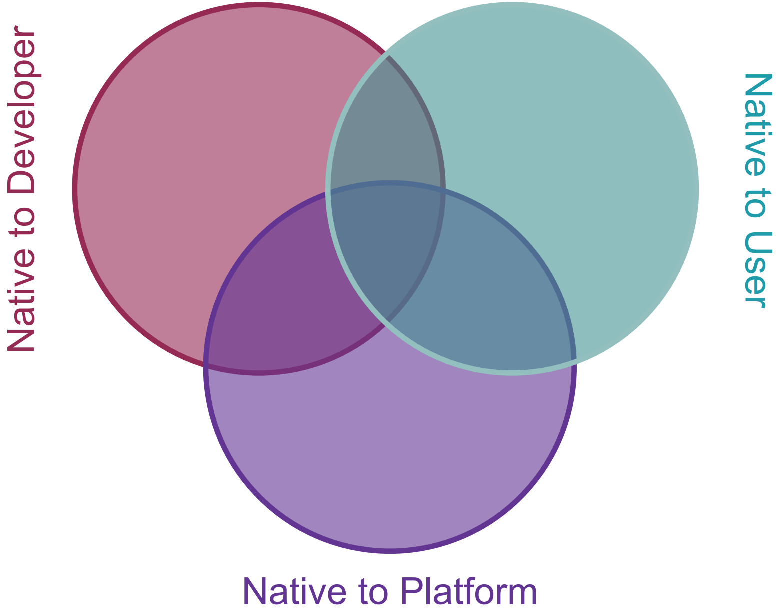
Changing Perspectives:



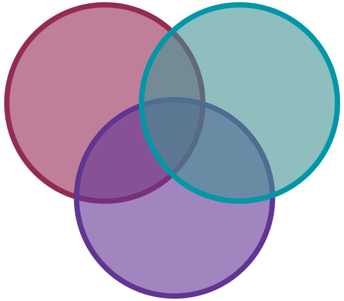
Changing Perspectives:



Changing Perspectives:

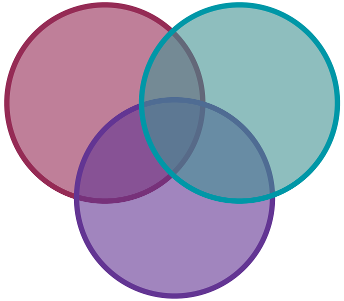


Changing Perspectives: Web



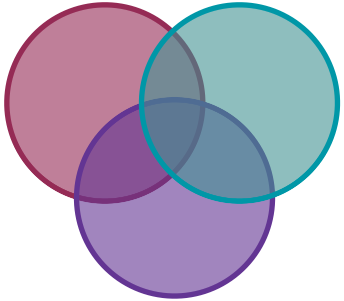
Technology	Native to Developer	Native to Platform	Native to User
Xamarin	○	○	●
React Native	●	●	●
Flutter	○		
KMP	○	●	●

Changing Perspectives: Android



Technology	Native to Developer	Native to Platform	Native to User
Xamarin		●	
React Native		○	
Flutter	○	●	○
KMP	●	●	●

Changing Perspectives: iOS



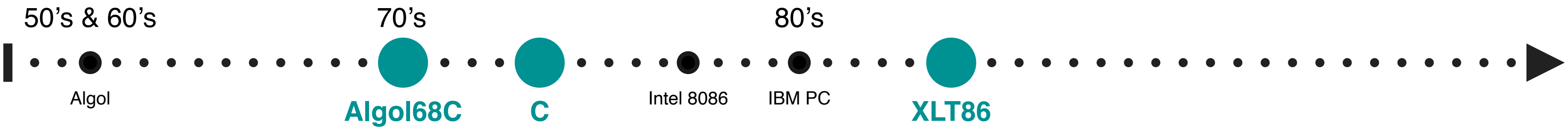
Technology	Native to Developer	Native to Platform	Native to User
Xamarin		●	
React Native		○	
Flutter		●	○
KMP	○	●	●

Changing Perspectives: iOS (near future)

Technology	Native to Developer	Native to Platform	Native to User
Xamarin		●	
React Native		◐	
Flutter		●	◐
KMP	●	●	●

Mapping the Quest through Time

CPU Era



50’s & 60’s

70’s

80’s

Algol

Algol68C

C

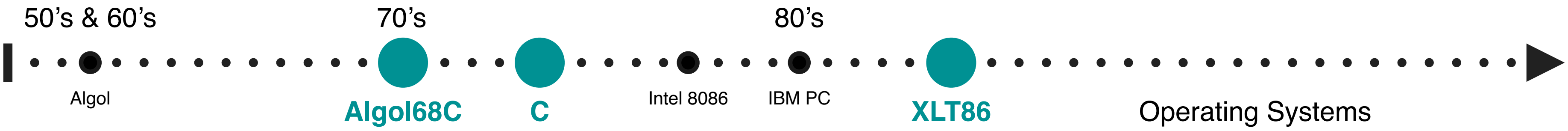
Intel 8086

IBM PC

XLT86

🟩 Fully Native Multiplatform

OS Era



50’s & 60’s

Algol

70’s

Algol68C

C

Intel 8086

80’s

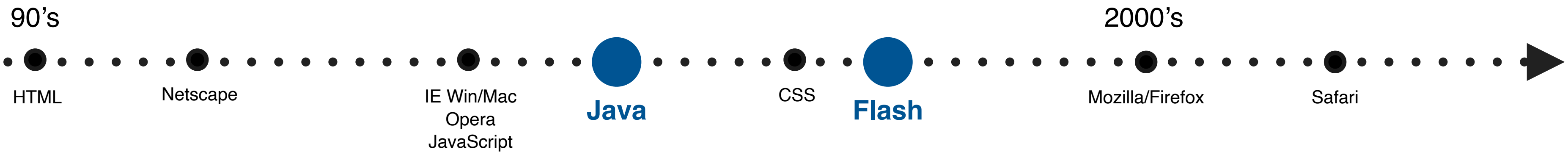
IBM PC

XLT86

Operating Systems

🟩 Fully Native Multiplatform

Web Era



90's

HTML

Netscape

IE Win/Mac
Opera
JavaScript

Java

CSS

Flash

2000's

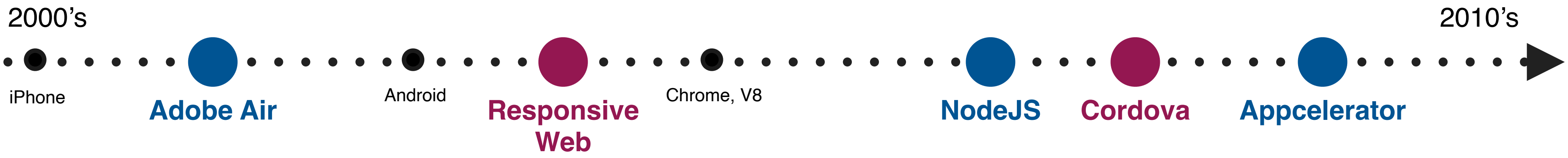
Mozilla/Firefox

Safari

🟦 Partially Native Multiplatform

🟩 Fully Native Multiplatform

Mobile Era



2000’s

iPhone

Adobe Air

Android

Responsive Web

Chrome, V8

NodeJS

Cordova

Appcelerator

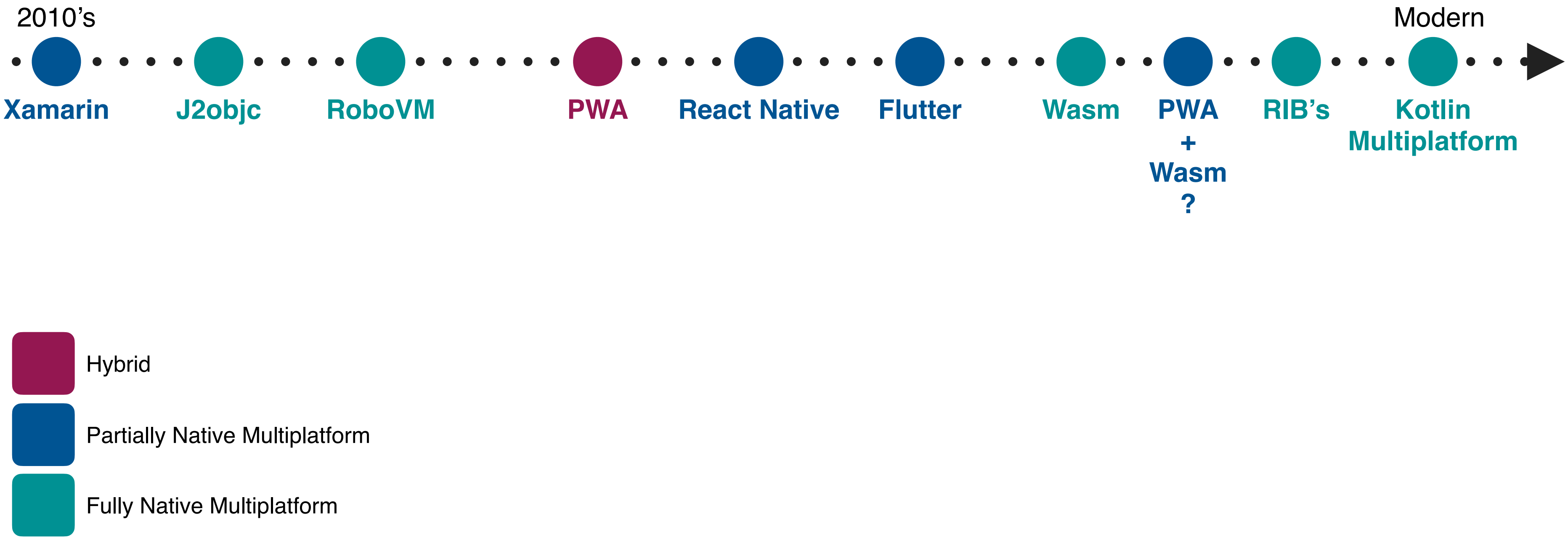
2010’s

🟪 Hybrid

🟦 Partially Native Multiplatform

🟩 Fully Native Multiplatform

Modern Era



2010's

Modern

Xamarin

J2objc

RoboVM

PWA

React Native

Flutter

Wasm

PWA
+
Wasm
?

RIB's

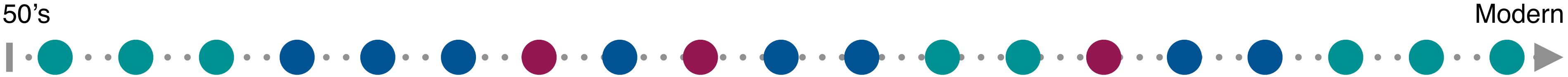
Kotlin Multiplatform

🟪 Hybrid

🟦 Partially Native Multiplatform

🟩 Fully Native Multiplatform

Mapping the Quest through Time



🟪 Hybrid

🟦 Partially Native Multiplatform

🟩 Fully Native Multiplatform

Kotlin Multiplatform

Efficient Developers

More Features

Fewer Bugs

Reach all the Users

Kotlin Multiplatform

Lower Performance

Slower Innovation

Poor UI

Vendor Lock-in

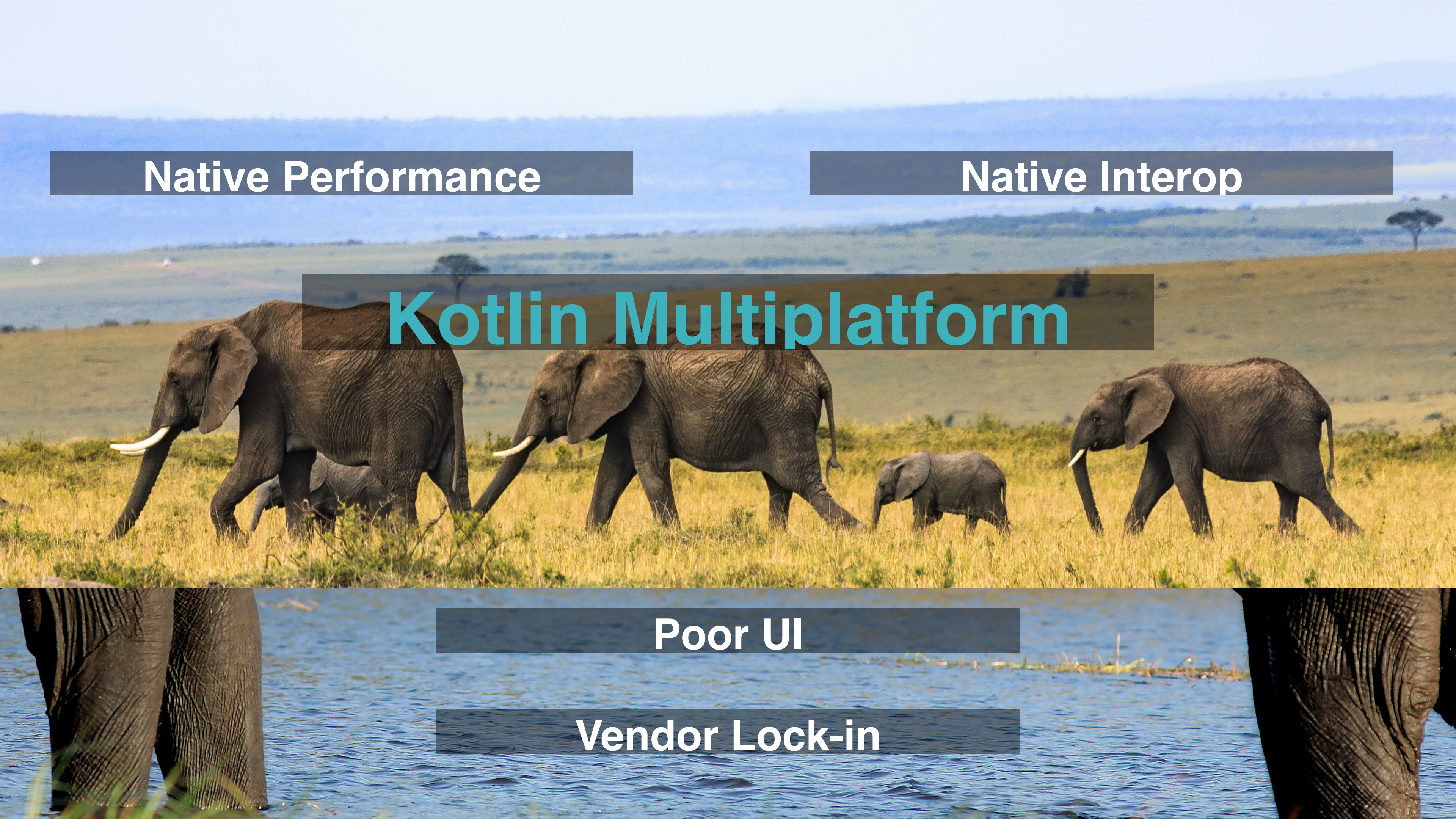
Native Performance

Kotlin Multiplatform

Slower Innovation

Poor UI

Vendor Lock-in



Native Performance

Native Interop

Kotlin Multiplatform

Poor UI

Vendor Lock-in

Native Performance

Native Interop

Kotlin Multiplatform

Native User Experience

Vendor Lock-in

Native Performance

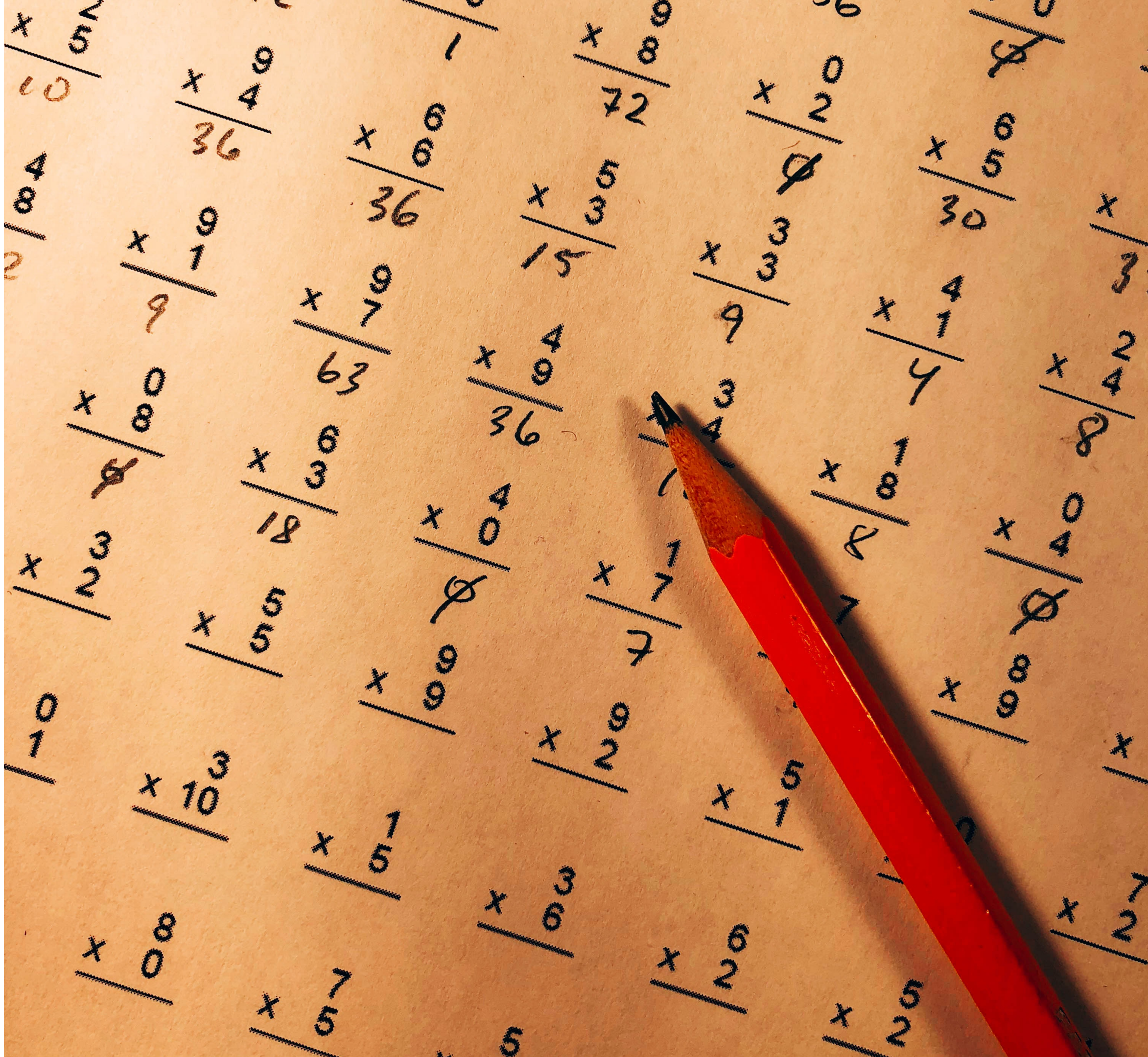
Native Interop

Kotlin Multiplatform

Native User Experience

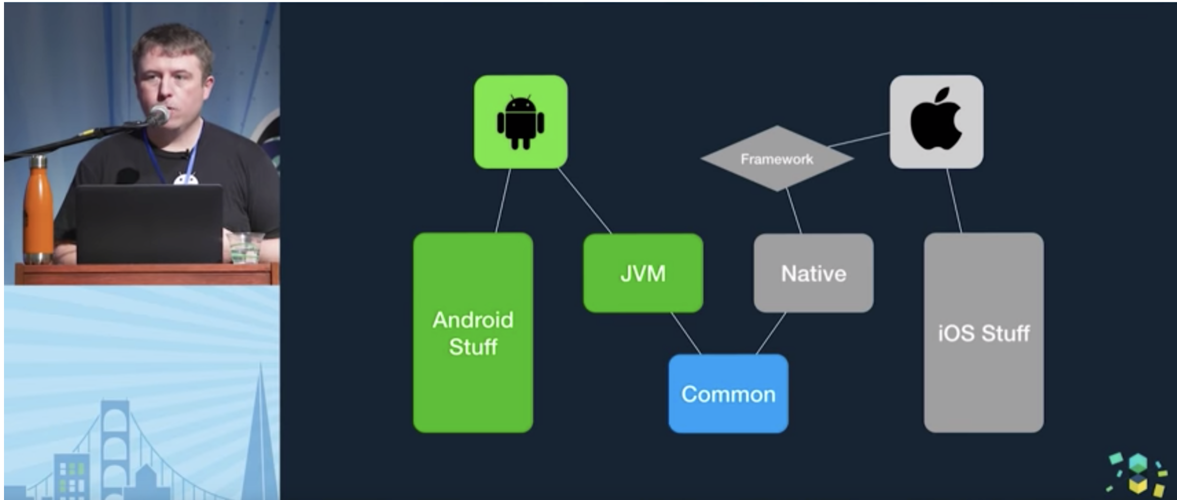
Native Dev Experience

Your Homework



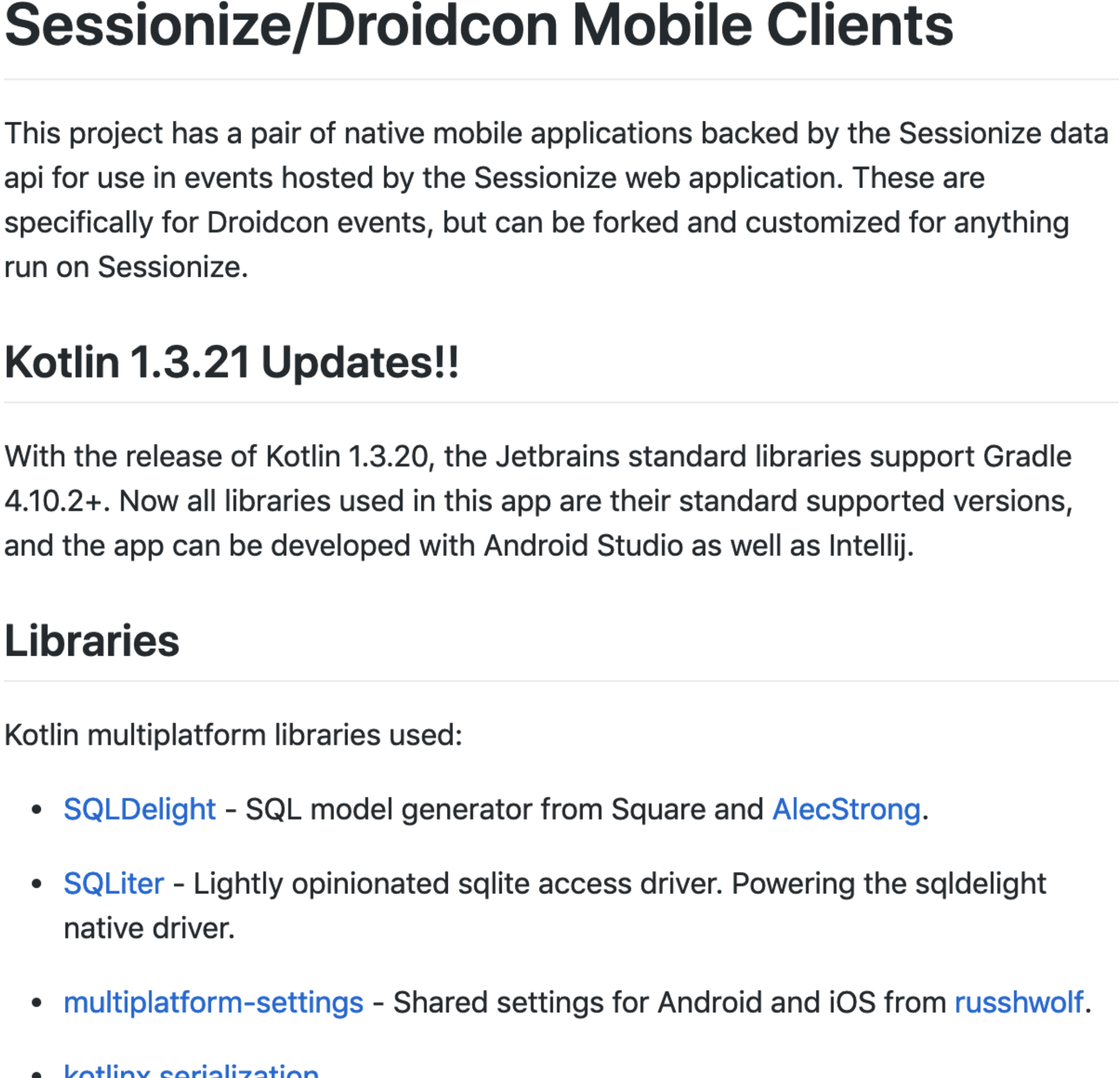
Your homework

- Watch related conference talks



Your homework

- Watch more conference talks
- Clone some projects



Sessionize/Droidcon Mobile Clients

This project has a pair of native mobile applications backed by the Sessionize data api for use in events hosted by the Sessionize web application. These are specifically for Droidcon events, but can be forked and customized for anything run on Sessionize.

Kotlin 1.3.21 Updates!!

With the release of Kotlin 1.3.20, the Jetbrains standard libraries support Gradle 4.10.2+. Now all libraries used in this app are their standard supported versions, and the app can be developed with Android Studio as well as Intellij.

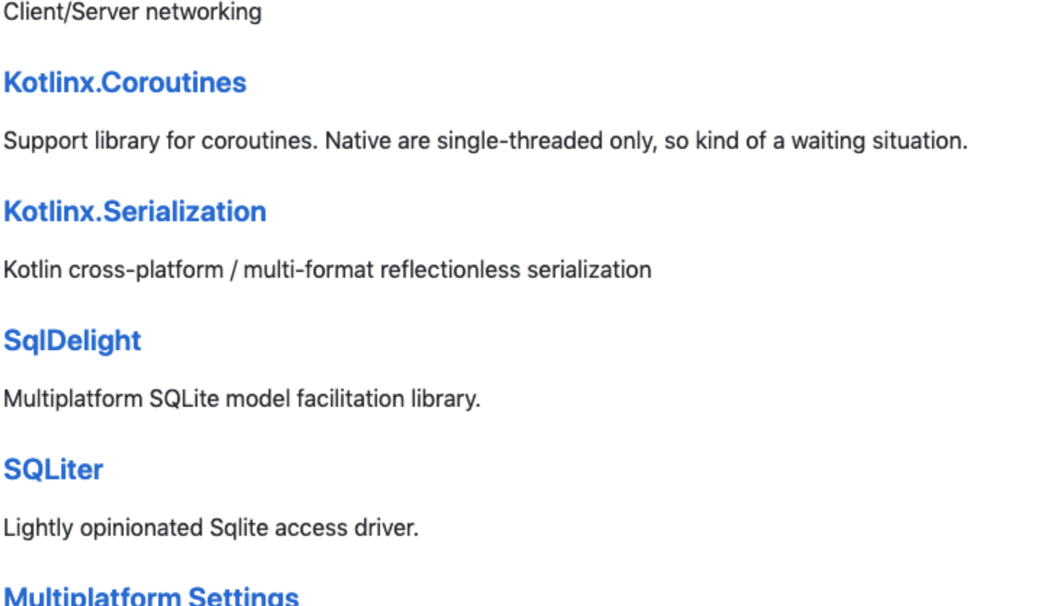
Libraries

Kotlin multiplatform libraries used:

- SQLDelight - SQL model generator from Square and AlecStrong.
- SQLiter - Lightly opinionated sqlite access driver. Powering the sqldelight native driver.
- multiplatform-settings - Shared settings for Android and iOS from russhwolf.
- kotlinx serialization

Your homework

- Watch more conference talks
- Clone some projects
- Contribute to and be supported by the community



Client/Server networking

Kotlinx.Coroutines

Support library for coroutines. Native are single-threaded only, so kind of a waiting situation.

Kotlinx.Serialization

Kotlin cross-platform / multi-format reflectionless serialization

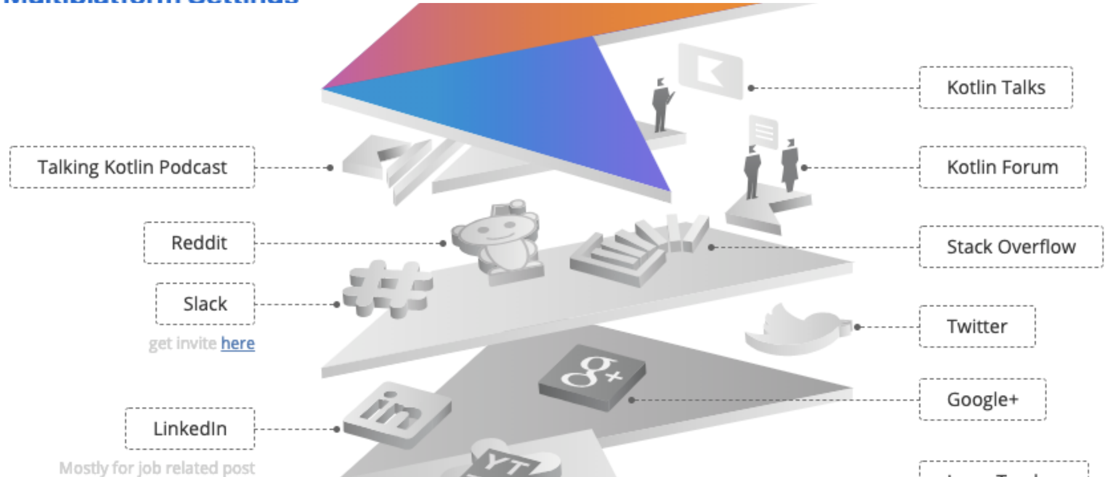
SqlDelight

Multiplatform SQLite model facilitation library.

SQLiter

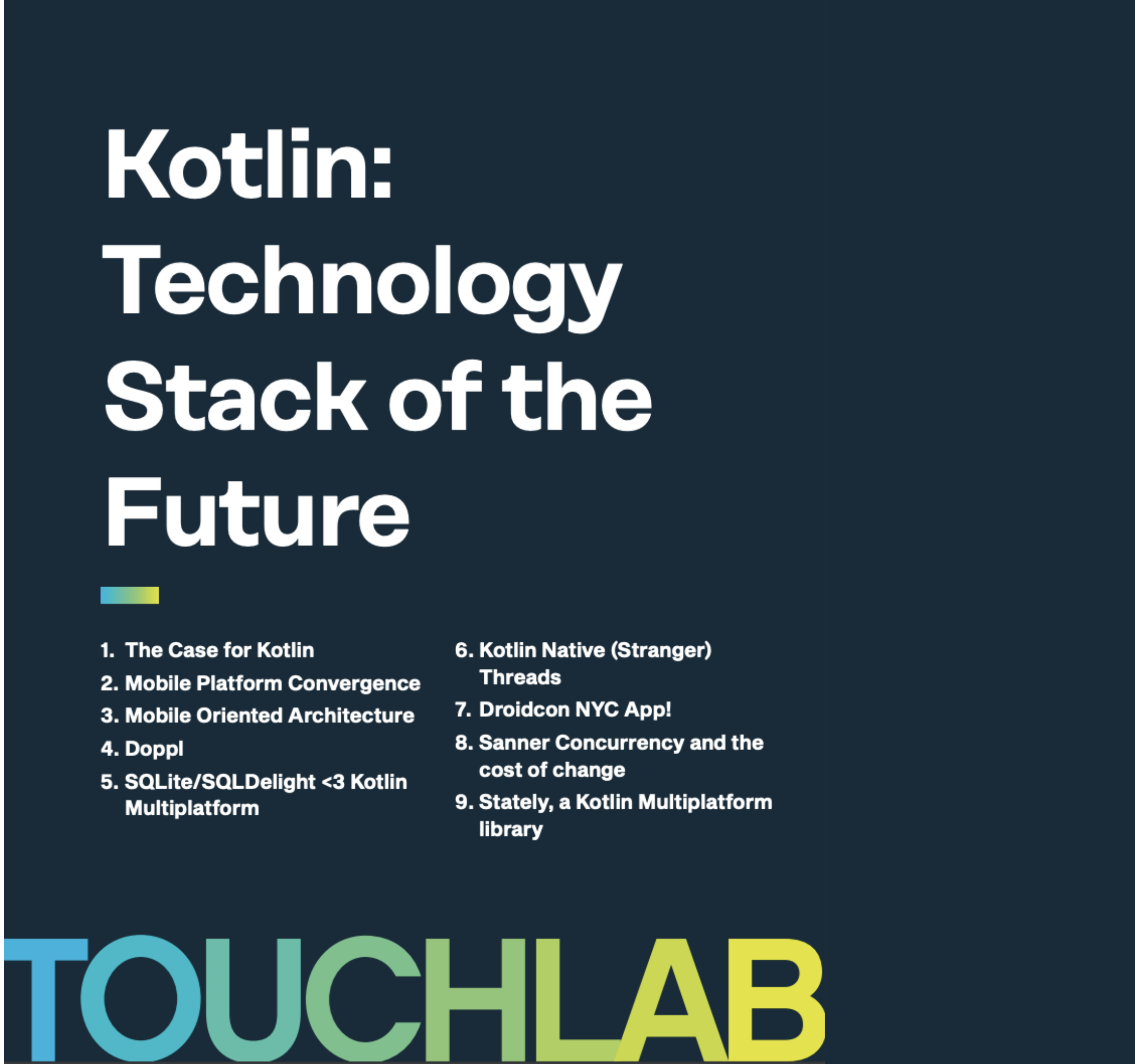
Lightly opinionated Sqlite access driver.

Multiplatform Settings



Your homework

- Watch more conference talks
- Clone some projects
- Contribute to and be supported by the community
- Talk to Touchlab



Kotlin: Technology Stack of the Future

1. The Case for Kotlin
2. Mobile Platform Convergence
3. Mobile Oriented Architecture
4. Doppl
5. SQLite/SQLDelight <3 Kotlin Multiplatform
6. Kotlin Native (Stranger) Threads
7. Droidcon NYC App!
8. Sanner Concurrency and the cost of change
9. Stately, a Kotlin Multiplatform library

TOUCHLAB