MTAT.03.262

Multi-platform Mobile App. Development with Apache Cordova

Jakob Mass
jakob.mass@ut.ee
Introduction

• Fragmented market
  • Mobile market has recently been stable, but lines are getting blurred: wearables, TVs, IoT, ...

• Developers have limited resources

<table>
<thead>
<tr>
<th>Platform</th>
<th>Market Capture*</th>
<th>Language</th>
<th>IDE</th>
<th>Market Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>88%</td>
<td>Java, Kotlin</td>
<td>Android Studio</td>
<td>Google Play Store</td>
</tr>
<tr>
<td>iOS</td>
<td>11.9%</td>
<td>Objective-C, Swift</td>
<td>Xcode</td>
<td>iTunes App Store</td>
</tr>
<tr>
<td>Others (Windows, Blackberry, ..)</td>
<td>&lt; 0.1%</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

*2018 Q1 according to Statista
Cross-Platform Development Principle

• Alternative to native app development
• Cross-Platform Tools (CPT)
• Share a code base between platforms
  • As large as possible

*Develop Once, Run Anywhere*

• Decrease dev. and maintenance cost of applications
• Can you name any multiplatform technologies you are already using daily?
Web-based CPT

The most common CPT approach.
- App UI with HTML + CSS
- Functionality with JavaScript

Two strategies:

1. “Normal webpage”
   - Load the app. from a remote webserver

2. Hybrid Application
   - Webpage contained locally on the device, using a web-to-native wrapper
   - Uses Platforms built-in browser engine
   - Can be distributed on the platforms application market
Other CPT approaches

• Interpreted Approach
  • e.g. Titanium
  • A dedicated interpreter on the device executes non-native code

• Cross compiled
  • e.g. Xamarin, Mono
  • Non-native code is compiled to native during build

• Others
### Table 1
Resume of different cross platform approaches.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Programming language</th>
<th>Supported platforms</th>
<th>Pros</th>
<th>Cons</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>HTML, CSS, Javascript</td>
<td>Android, iOS, Windows, BlackBerry&lt;sup&gt;a&lt;/sup&gt;</td>
<td>– Easy to update</td>
<td>– No access to application store</td>
<td>jQuery mobile,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– No installation</td>
<td>– Network delays</td>
<td>Sencha Touch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Same UI over different devices</td>
<td>– Expensive testing</td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td>HTML, CSS, Javascript</td>
<td>Android, iOS, Windows, BlackBerry,</td>
<td>– Access to application store&lt;sup&gt;b&lt;/sup&gt;</td>
<td>– No native UI</td>
<td>PhoneGap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Support to most smartphone hardware</td>
<td>– No native look and feel</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Access to application store</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Native look and feel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interpreted</td>
<td>Javascript</td>
<td>Android, iOS, Blackberry</td>
<td>– Native UI</td>
<td>– Platform branching</td>
<td>Titanium</td>
</tr>
<tr>
<td>Cross-Compiled</td>
<td>C#, C++, Javascript</td>
<td>Android, iOS, Symbian</td>
<td>– Real native application</td>
<td>– Interpretation step</td>
<td>Mono, MoSync</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– UI non reusable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– High code conversion complexity for complex applications</td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup> Support depends on the browser chosen by the user.

<sup>b</sup> Apple store usually tends to refuse applications developed with this approach [34].

Which one to choose?

Over 100 existing CPT-s! [2]
• Development time consumption, learning curve
• How many platforms are supported?
• Feature/functionality support
  • Gestures
  • Native API-s, e.g. sensors
• Application performance, UI responsiveness
• UX, conformance to “Native design experience”


Apache Cordova

aka Phonegap

• Open-source

• HTML5, CSS3 and JavaScript
  • You can use web dev. libraries such as AngularJS, jQuery, jQuery Mobile
  • Numerous community-create Cordova plugins

• Applications
  • Execute within wrappers (containers) targeted to each platform
  • Rely on standards-compliant API bindings to access device capabilities such as sensors, data, network status, ..

• Large community

• Contributors of Cordova project include Adobe, Blackberry, Google, IBM, Intel, Microsoft, Mozilla, ...
Cordova Platform Support

Version 8.X:

Version 7.X:

Available platforms:
- android ~6.3.0
- blackberry10 ~3.8.0 (deprecated)
- browser ~5.0.0
- ios ~4.5.1
- osx ~4.0.1
- ubuntu ~4.3.4 (deprecated)
- webos ~3.7.0
- windows ~5.0.0
- www ^3.12.0

Web App

Project files
- HTML, JS, CSS, Resources

config.xml
- Information about the app
  - Cordova features, plugins used
- Application behaviour e.g. whether it handles orientation changes

Plugins

Provide an interface for Cordova and native components to communicate, bindings to standard device APIs

• Core Plugins
  • Maintained by the project

• 3rd party plugins

Plugins

Provide an interface for Cordova and native components to communicate:

- Core Plugins
  - Maintained by the project
- 3rd party plugins

Cordova Development Workflows

2 approaches

1. Cross-platform workflow
   • Minimize platform-specific development
   • CLI tool automates a lot of tasks, such as generating binaries for each platform
   • Recommended, unless you have a specific need for platform-oriented functions

2. Platform-centered workflow
   • Gives more control per-platform
   • Enables mixing custom native components with web-based Cordova components
   • As a downside, you have to manage build cycles of the platform

In the next examples, we are following the first, CLI tool-based approach
Cordova Dev. Environment

Prerequisites:
• Node.js and NPM installed
• Cordova CLI:
  • npm install -g cordova
• SDKs of the platforms you want to build for:
  • Android SDK
    • As of 24.10.2018, Cordova supports up to Android API v27
    • Android Emulator for testing
  • iOS SDK + Xcode
    • only runs on the OS X operating system!

For more details:
Create an Android App

```bash
$cordova create myapp com.example.hello MyApp
$cordova platform add android
$cordova build android
$cordova run android
```

Re-creating HA1

• Slightly simplified
• Display contacts, send e-mails

On Android:

We need some API functions (Contacts, E-mails), so we need plugins which provide those

$ cordova plugin add cordova-plugin-contacts
$ cordova plugin add cordova-plugin-email

https://www.npmjs.com/package/cordova-plugin-email
• Demo
..on iOS
Other Tools

• Ionic – platform based on Cordova
  • Includes a lot of useful tools, libraries
    • AngularJS, Cloud Designer, CI

• Adobe Phonegap
Summary

• Great for UI-driven applications without complex, smartphone-oriented functionality
• Proficiency with web technologies is a must
• Even with the multi-platform workflow of Cordova, always be alert for platform-specific quirks
  • Not all plugins behave the same on all platforms!
• CPT-s are still an evolving scene
  • Ionic