Cross-Platform Development:

Target More Platforms and Devices with a Minimal Amount of Source Code

What is cross-platform development?

- **Cross-platform development** produces a single code base that can be run across multiple operating systems such as Microsoft Windows plus Mac/Apple and/or Android.

Cross-platform applications can reach your customers through whatever device they have in front of them, wherever they are: using a Windows* PC at work, an Apple* iPad* at the local coffee shop or an Android* phone on the go.

Advantages

- Consolidated Development Effort – Using a single API /code base saves time on development, localization and maintenance of the code base.
- Decreased Development Time – Cross-Platform tools (CPTs) not only provide one tool for multiple OSs, they can also simplify development with dynamic and comprehensive libraries.
- Code and Asset Sharing – Most of the major frameworks have official repositories for sharing code and assets or purchasing them from other developers/vendors.
- Fewer Skill Sets – The required technical skills sets can be focused around a single API to implement and support the project.

Disadvantages

- SDK Lock In – Once you commit to a framework, you are dependent on the tool set and the availability of development resources who support it.
- Outdated API Risk – Cross-platform SDKs may lag behind the latest and greatest features offered by native SDKs possibly making applications appear less up-to-date and current.
- Testing – Testing cross-platform apps may be more complicated since different platforms can exhibit slightly different behavior and subtle bugs.
- Limited Feature Set – Cross-platform SDKs are an abstraction of the various platforms into one uniform interface; this abstraction may limit the feature set compared to the native API.

CPT Challenge: Dynamically Adapting to Screen Parameters

- **Truly cross-platform** means that the application provides a good user experience across a variety of screen sizes, resolutions, aspect ratios and orientations.
  - Scaling – Scaling components and fonts according to the screen dimensions is the simplest approach and optionally, adjust padding around UIT components to accommodate various aspect rations.
  - Responsive Images and Layout – Images may be cropped or optimized for different resolutions, also consider a fluid layout; rearrange sections based on available screen real estate.
  - Managing “Idiosyncrasies” – For example, all OSs have differences in the behavior of platform runtimes that complicate cross-platform development.

Top 5 Cross-Platform Tools*

- Qt
- Adobe AIR
- Appcelerator
- Apache Cordova
- Sencha

* Developer Economics 2013 Q1 Report

Qt ("Cute")

- **Qt** targets a number of embedded, desktop and mobile platforms. Developers write using "QML", a CSS and JavaScript like language. Apps are backed by an extensive set of C++ libraries and utilize graphics/UI components written in C++.

Qt (“Cute”)

Pros

- Provides a substantial set of libraries containing intuitive APIs for features like threading, networking, animations and more.
- Qt’s IDE tooling (Qt Creator IDE & Qt Designer) are solid development tools, and code profiling is available in QML Profiler.
- Qt Linguist enables translation and internationalization in applications – providing the support of multiple languages within an app (in a single binary).

Cons

- Qt’s tools are advertised as a “complete tool chain”, and QML is a proprietary language specific to Qt’s stack. Committing to this approach could be viewed as ‘platform lock-in’, and may limit the re-use of existing developer skill sets when adopting a CPT.
- Pricing of the toolset /support may be higher than support cost for other CPTs.

Adobe Air

- **Adobe Air** allows developers to combine HTML, JavaScript, Adobe Flash® and Flex technologies, and ActionScript® + to deploy Rich Internet Applications (RIAs) on a broad range of devices including desktop computers, netbooks, tablets, smartphones, and TVs.

Adobe Air

- **Pros**
  - Adobe AIR has impressive reach – running on a wide array of desktop and mobile devices. In addition, if a more involved/animated UI is required (and a native approach is not utilized), using AIR over a HTML/JavaScript/CSS approach may work well.
  - Most Flash/ActionScript developers consider the IDE tooling for these technologies as mature.
- **Cons**
  - The “elephant in the room” for many mobile developers is the fact that Adobe purchased Nitobi (and the rights to the PhoneGap name), clearly signaling to many that AIR may not be a long term strategy for mobile development. This combined with the rapid decline of Flash erodes the confidence many developers might otherwise have in choosing AIR.

Appcelerator

- **Appcelerator's Titanium** provides a unified (across devices) JavaScript API, coupled with native-platform-specific features. Developers write JavaScript and utilize a UI abstraction (the Alloy MVC framework) that results in the use of native UI components, greatly aiding UI performance compared to other hybrid options.

Appcelerator

- **Pros**
  - The use of native UI components is a performance win, and the Alloy framework attempts to normalize UI across platforms.
  - The use of JavaScript to normalize code across platforms enables you to leverage existing skills on multiple target platforms.
  - Appcelerator provides value-adds such as a Backend-as-a-Service (BaaS), app analytics and a marketplace for 3rd party components.
- **Cons**
  - Normalizing the UI across platforms, while arguably a “pro”, is also a “con” in that the SW development team will need to have experience on a proprietary technology (skills that are not directly transferrable outside Titanium).
  - Developers are required to manage target platform SDKs locally.

Apache Cordova

- **Apache Cordova / PhoneGap** supports creation of mobile apps using HTML, JavaScript and CSS. These assets run in “WebView” inside a native app container on the target platform. It is, conceptually a web app packaged within a native app container where JavaScript has access to device-level APIs that normal web apps would not access.

Apache Cordova

- **Pros**
  - A significant number of developers have experience with HTML, JavaScript and CSS. Apache Cordova allows developers to immediately leverage these existing skills.
  - Cordova apps install like a native application, and are able to leverage app store discoverability. Cordova follows a plugin architecture, which means that access to native device APIs can be extended in a modular way.
  - Cordova is open source *and* free, so there are no licensing costs.
- **Cons**
  - The performance of Cordova apps has often been criticized.
  - Cordova is “bare bones” and maybe incomplete. The open source/ plugin architecture works well if the plugins needed are available *or* if your web developers are capable writing their own custom plugin(s) as needed.

Sencha

- **Sencha Touch** is an HTML5 mobile app framework for building web apps that look and feel like native apps. Apps built with Sencha Touch can be used with Apache Cordova or Sencha’s native packager – either will package the app in a native container and enable access to device-level APIs unavailable to traditional web apps.

Sencha

- **Pros**
  - Includes a large set of interoperable products, from “Sencha Architect” (a visual HTML5 app builder) and “Sencha Touch Charts” (for data visualization) to IDE integration with Sencha Eclipse.
  - Sencha Touch offers a library of UI components, an extensible API and UI themes among other features.
  - Native packaging is possible via Apache Cordova/PhoneGap or Sencha’s SDK.
- **Cons**
  - Mobile apps written with Sencha Touch *can* exhibit performance issues if developers aren’t disciplined in writing efficient JavaScript and DOM structure(s).
  - Sencha’s emphasis on its own stack is perceived as vendor lock-in. Many developers already have established experience with preferred frameworks for HTML5/JavaScript/CSS based apps.

Conclusion

- **Cross-platform development** is the way to go if it fits both the application requirements and the skill set of the resources.

Carefully select the development tool/ SDK to make certain it supports all the required functionality of the application. Many of the frameworks support extensions that will allow the addition of native modules to fill gaps. Ultimately a cross-platform framework should reduce the development time-line and budget compared to a “multiple code base approach” developed with native frameworks.

Thank you

Created by: ASHVINS Group, Inc.
Email : sales@ashvinsgroup.com
www.ashvinsgroup.com

ASHVINSGROUP
TECHNOLOGY PROFESSIONALS