## Cross Platform Tools

* What is a cross platform tool?
  + 2 Types:
    - Native Compilation
      * Tools that allow for the compilation on multiple platforms. The app produced will be a native app for each individual platform.
    - Hybrid Apps
      * Usually, for mobile devices, this involves a web view wrapped in a native app. In other words, the native app simply displays a web view control, and the web view control shows a web interface that runs via the browser.
      * Often, through the use of the native app, the web view can access some of the device features such as contacts, gps, battery, etc.
      * This also includes interpreted language apps (not common).
  + In theory, the tool allows writing code once that will be then used to create apps for each platform. This is often difficult to achieve perfectly in practice due to quirks on each platform.
* Why develop using a cross platform tool? Advantages and Disadvantages
  + + Learn one tool, deploy to multiple platforms
  + + If the tool is robust and kept up to date, no need to learn new platforms as they emerge.
  + + Fast prototyping of simple apps, use of template techniques
  + – Quick pace of change may still pose a challenge to developers to “keep up”
  + - Lowest common denominator, often missing features or quirks in one platform will impact all other platforms as the developer may want to keep the app standard across all platforms.
  + – Debugging may be difficult on platforms that do not have the necessary features for the tool’s debugger.
  + - Might be best to just create custom apps to target platforms, though this is another trade off as now expertise must be maintained in multiple platforms.
* Popular cross platform tools
  + Cordova / Phonegap – Open source cross platform tool. Free cloud build service
  + Xamarin – C# based tool for developing cross platform apps – expensive
  + Ionic Framework – Hybrid mobile app tool originally built on Cordova but now are built on Angular
* Cordova / PhoneGap and Hybrid Apps
  + Brief history
    - Initially created by Nitobi, purchased by Adobe in 2011
    - Adobe released open source version named Apache Cordova later
    - Supports to some extent Android, Windows Phone, iOs, Blackberry, Ubuntu, Firefox OS, Fire OS, Windows Desktop
  + How they work under the hood
    - Hybrid apps, not truly native nor completely web based.
    - Layout rendering is done via web views, but wrapped as applications and packaged for app distribution
    - Have access through libraries and plugins to many native device APIs
    - Called Phonegap, Cordova, or sometimes Apache Callback
    - Several tools are built on top of PhoneGap including Ionic and Intel XDK
  + Online build service
    - Build.phonegap.com
    - Allows building for iOS (with dev license), windows 8, and android
    - Other platforms must be built from the command line
  + Plugins
    - There are over 1000 plugins at the plugin database online.
    - Support for plugins is not always complete for all platforms. Android and iOS are usually the best supported, by far.
* Xamarin
  + Brief History:
    - Team that created Mono (an open source .NET framework port) founded Xamarin.
    - Originally had a separate IDE called Xamarin Studio, which has mostly been deprecated.
    - Dec 2012 Xamarin.Mac allowed building of C# applications for OS X
    - Xamarin 3 – Xamarin.Forms allows portable controls that are mapped to native controls of Android, iOS, and Windows Phone (main focus of our work in this class)
    - Feb 2016 Microsoft acquired Xamarin and released the previously expensive toolset for free and began work on tighter integration with Visual Studio
  + How a Xamarin project works
    - Solution is generated with multiple projects within it:
      * Shared Project – Majority of code goes here, everything that isn’t platform specific.
      * iOS Project – iOS specific code for accessing device features or dealing with quirks of iOS.
      * Android Project – Same as iOS project but for Android.
      * Windows UAP project – not used in this class
    - Debugging for Android: Uses USB or emulator and the Android SDK
    - Debugging for iOS: Uses a remote agent on a Mac
  + Output platforms
    - Outputting and building for Android works well and uses the Android SDK tools. The Visual Studio environment handles images, API levels, etc. An APK is packaged and deployed to the device, or can be outputted for the store.
    - In order to circumvent Apple’s restrictive development rules, Xamarin takes an innovative approach by allowing development on a Windows machine and compiling and deployment on a remote Mac system.
      * Xamarin Agent and Local User on the Mac must have local admin privileges
      * Requires current versions of XCode and the Xamarin Mac software
      * Log the user in locally on the Mac, then connect to the session from Visual Studio
      * The remote agent can deploy to the iOS simulator, which is visible on the Mac.