



Visual Studio Tools for Apache Cordova

Presented by Mel Leeb @ Bennett Adelson



Visual Studio Tools for Apache Cordova

Presented by Mel Leeb @ Bennett Adelson

3 Years Ago ...

PhoneGap Presentation

The slide illustrates the PhoneGap workflow. At the top center is the HTML5 logo. To the left is the PhoneGap logo. Below these, a diagram shows a 'Single Code Base' (represented by two document icons) leading to a 'PhoneGap' box (represented by a cube), which then leads to 'Deploy to Multiple Platforms' (represented by three document icons). To the right of the diagram are images of a tablet and two smartphones, all displaying a jQuery mobile application interface. Below the diagram is the text 'm.YourCoolWebSite.com'. At the bottom, it says 'Presented by Mel Leeb' and the 'BENNETT ADELSON' logo with 'Microsoft® Solution Center' underneath.

HTML5

PhoneGap

Single Code Base PhoneGap Deploy to Multiple Platforms

m.YourCoolWebSite.com

Presented by Mel Leeb

BENNETT ADELSON
Microsoft® Solution Center

Agenda

- Have Fun
- Learn Some Cool Stuff
- Realize you already know how to code for mobile web apps
- See demos that you can download from Microsoft today and test on simulators and devices for: iPhone, Android, Windows Phone, etc.



Overview

- Tooling Support for Cordova Apps within Visual Studio
- Review Microsoft's Strategies for Mobile
- Xamarin vs. Cordova Apps
- Building & Debugging Cordova Apps
- Hybrid App Alternatives
- Additional Resources

What it takes to build a Cordova App

Dependencies to install:

1. Joyent Node.js
2. Google Chrome
3. Git Command Line Tools
4. Apache Ant
5. Oracle Java 7
6. Android SDK
7. SQLite
8. WebSocket4Net

Need to configure:

- Command line targets
- Environment variables
- Deployment details
- Build Server
- Web Server
- and more!

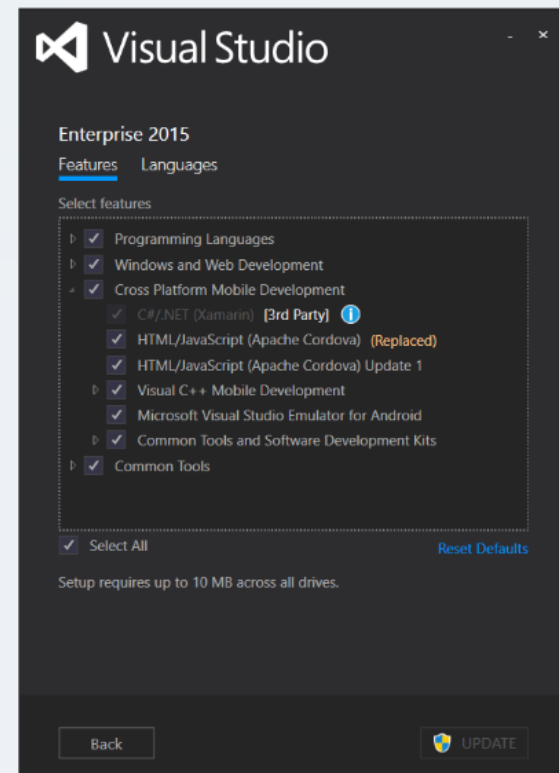
- A lot of what it takes to get up & running normally involves a number of steps and software to install
- A number of Cordova actions are often done via its command-line interface (CLI) (This tool allows you to create new projects, build them on different platforms, etc.)
 - While helpful, many developers prefer working within an IDE

Evolution of Mobile Tooling Support

- History:
 - Microsoft collaborated with PhoneGap as early as 2011, and now they are integrating Cordova into their tooling
 - Microsoft also partnered with Xamarin in 2013 and have expanded their offerings (ex. - Xamarin Templates for iOS and Android in Visual Studio 2015)
- Visual Studio Support for Cordova:
 - Added in VS 2013 Update 2
 - Starting with VS 2013 Update 4 - Tools Available

Visual Studio Integration

- Tooling for Cordova is now integrated with Visual Studio
 - VS 2013 - as separate download; or
 - VS 2015 - as part of the installation

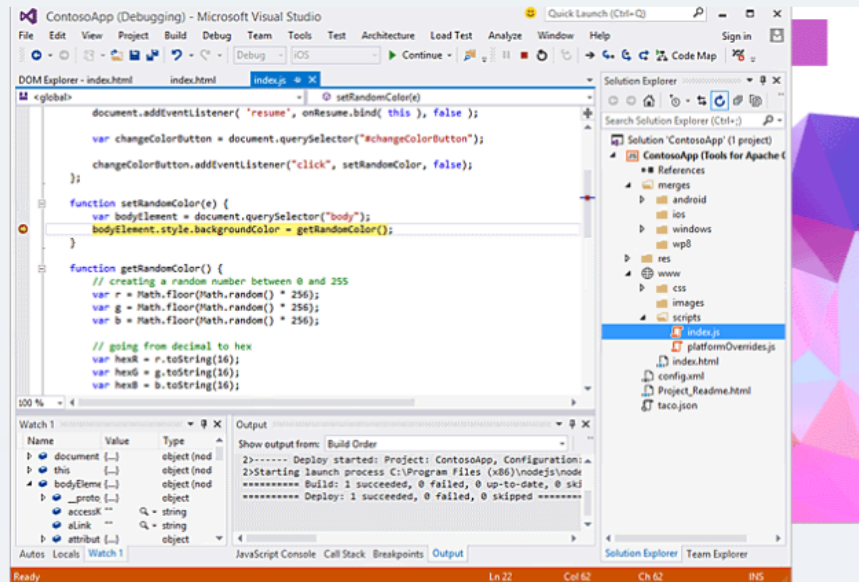


Supported Platforms



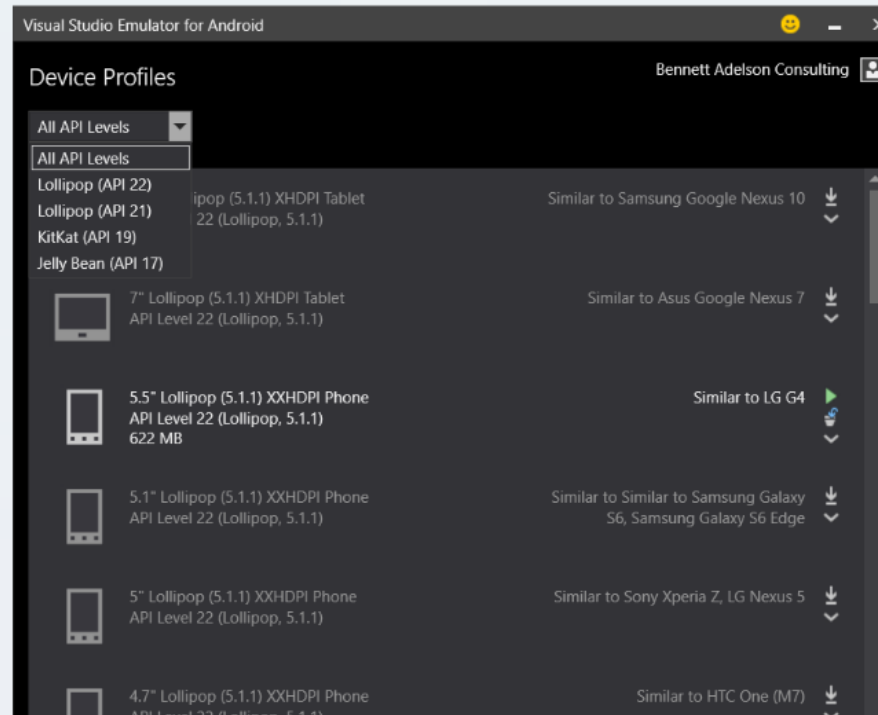
- You can build apps using the tools for these devices and platforms:
 - Android 2.3.3 and later (4.4 provides the best developer experience)
 - iOS 6, 7, and 8
 - Windows 8 and 8.1
 - Windows Phone 8 and 8.1
 - Windows 10.0

Debugging Support within Visual Studio



- Tightly integrated debugging support for Cordova apps
- Launch your app in any emulator or device (local or remote, including iOS) and debug locally all within Visual Studio
- New DOM Explorer window - F12 Developer tool-like support, now inside the VS IDE
- Update on save for Ripple – no need to rebuild! (i.e. - "Live Reload Server")

Fast Android Emulators

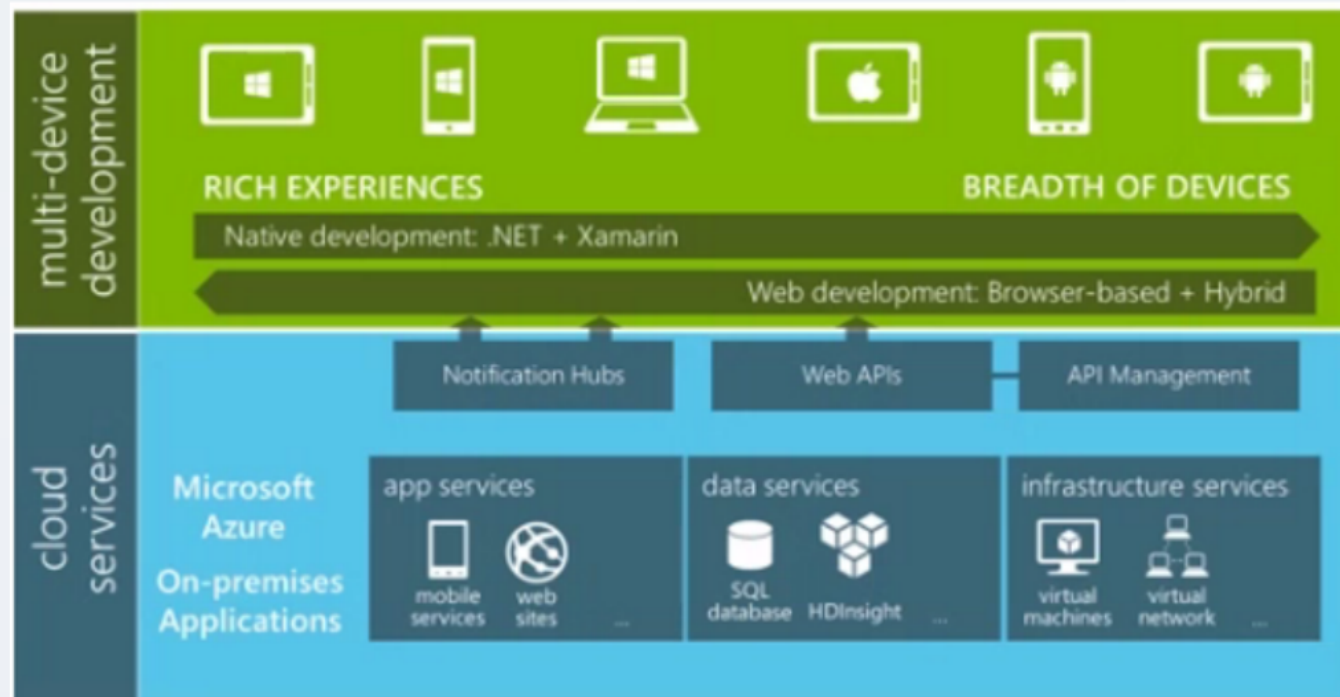


- Typically the original Android Emulators have been notoriously slow
- Alternative like Genymotion are popular, but costs \$ for a Business Edition license
- Microsoft has built and now includes a free and fast set of emulators for Android
- Supports various Android API levels and device types

Microsoft's Strategies for Mobile

Two General Approaches:

- **Xamarin**: Native development with VS, .NET, and Xamarin
- **Cordova**: (Hybrid) Web development with Apache Cordova and VS



Xamarin vs. Cordova

Choose strategy based on your business app's and dev team's goals & needs.

Power

- Xamarin - 100% access to native APIs. Native UI controls and performance.
- Cordova Hybrid App within Visual Studio - Access to 22 common APIs (ex. - Camera, Contacts, Geolocation, Accelerometer, etc.)

Cost

- Xamarin - Per platform licensing model. Licensing costs (up to \$3800 developer / year for both iOS and Android platforms).
 - 20% discount for MSDN subscribers.
- Cordova Hybrid App within Visual Studio - Free

* % Code Reuse

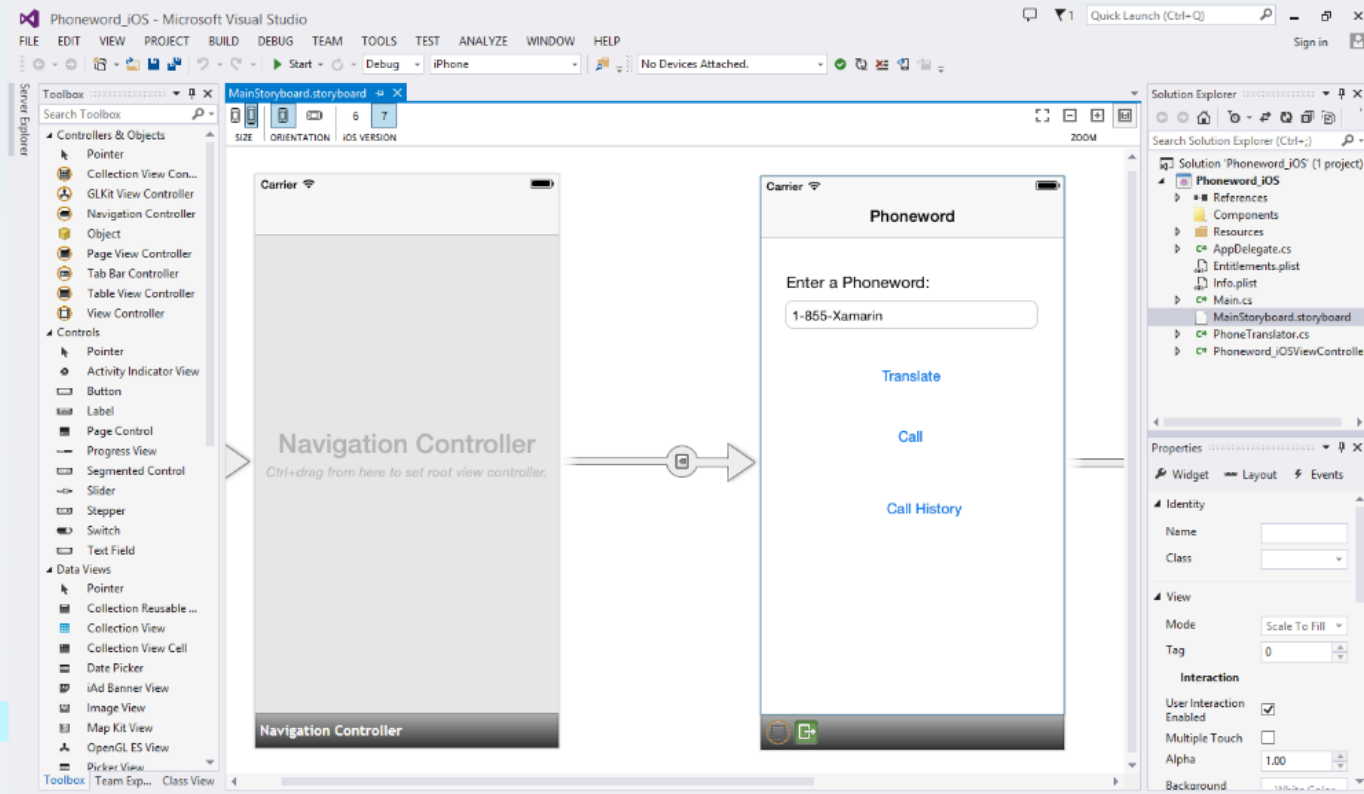
- Xamarin (Native) - Approx. 60-70% (Separate UI projects)
- Xamarin (Forms) - Approx. 80-90% (Separate UI projects, but shared UI code)
- Xamarin (Hybrid w/ Razor) - Approx. 90-95% (Separate UI projects, with just a little bootstrap code)
- Cordova Hybrid App within Visual Studio - Approx. 95-100% (Single code base)

Xamarin - Development

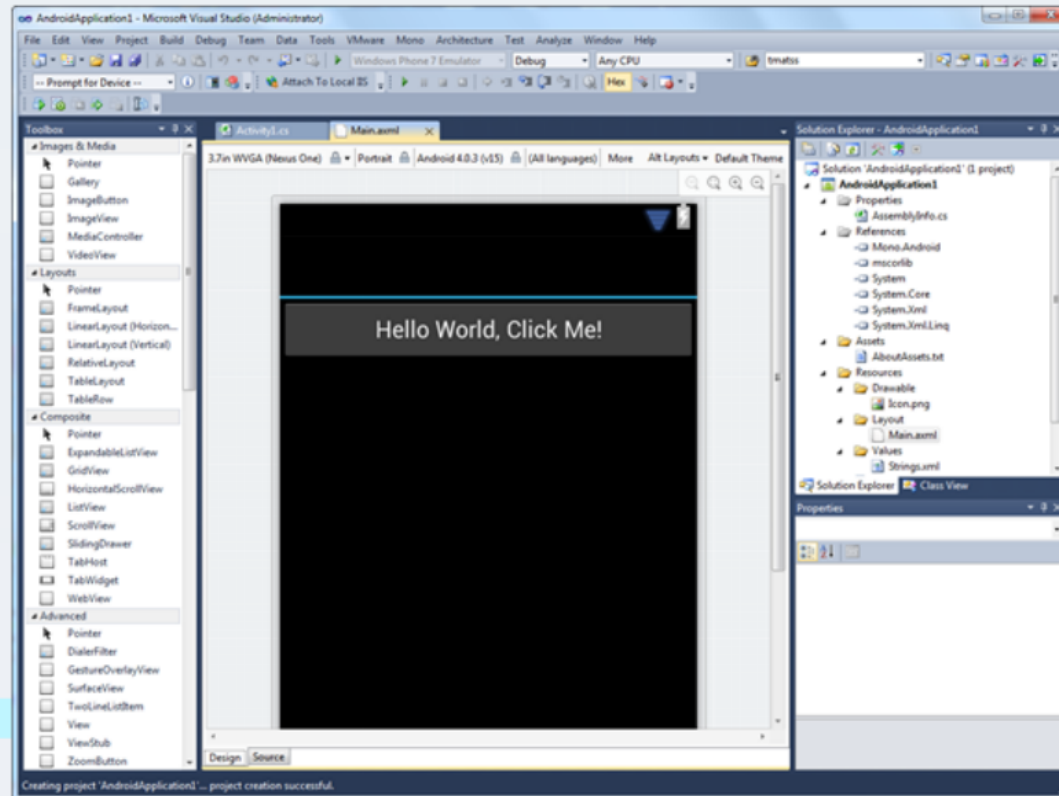
Create Rich Native apps for iOS, Android, and Windows while maximizing code reuse

- Fully native UI w/ Xamarin 100% of APIs are exposed, so whatever you can do in XCode a/ Objective-C, Android / Java, etc., you can do in C# and VS
- Xamarin Extension for VS also gives you the ability to create Android & iOS projects inside VS
- High degree of sharing comes from putting most of your business logic inside either a Shared Project or Portable Class Library (PCL) Project type

Xamarin - UI Designer for iOS

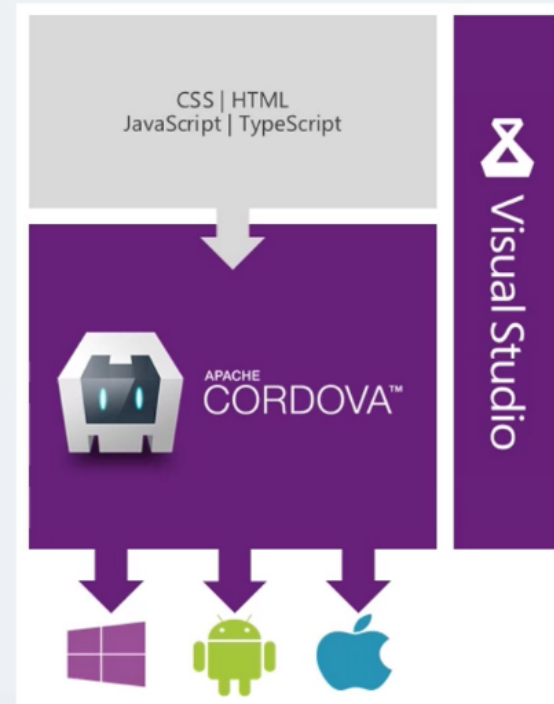


Xamarin - UI Designer for Android



Cordova Apps

- Create a single Hybrid app (project) that leverages a team's skillset in Web Standards: HTML, CSS, JS
- Flexibility to use any of the JavaScript frameworks you already know and love (AngularJS, WinJS, etc.)
- Native device access with support for common and custom plug-ins exposed as JavaScript APIs

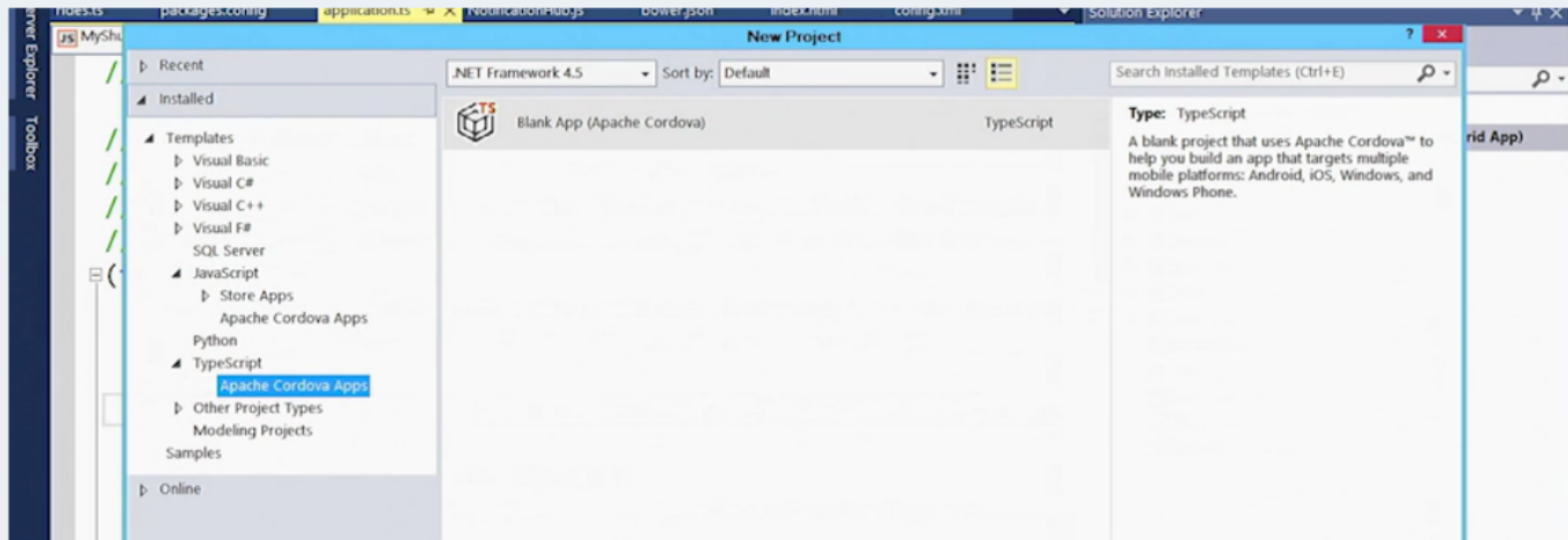




Visual Studio Tools for Apache Cordova

Presented by Mel Leeb @ Bennett Adelson

Cordova VS Templates

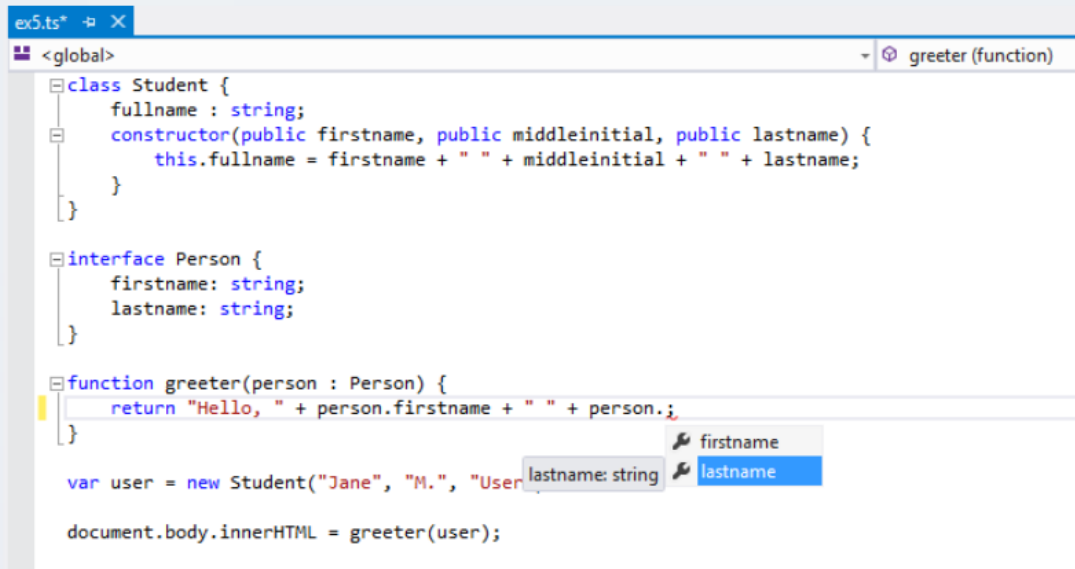


In VS, the Tools for Apache Cordova comes with templates for both JavaScript and TypeScript ...

- Microsoft recommends TypeScript for larger team Cordova projects
- TypeScript projects can also contain a mix of JavaScript files

TypeScript

- An extension to JavaScript, which is designed to support the productivity and scale of statically typed languages
- Offers classes, modules, and interfaces to help you build robust components
- These features are available at development time for improved design-time application development, yet are compiled into simple JavaScript files



```
ex5.ts*  X
<global>  greeter (function)
class Student {
  fullname: string;
  constructor(public firstname, public middleinitial, public lastname) {
    this.fullname = firstname + " " + middleinitial + " " + lastname;
  }
}

interface Person {
  firstname: string;
  lastname: string;
}

function greeter(person: Person) {
  return "Hello, " + person.firstname + " " + person.;
}

var user = new Student("Jane", "M.", "User", lastname: string, lastname);

document.body.innerHTML = greeter(user);
```

Hybrid Web UI Design Considerations

- Use a mobile-responsive cross-platform framework (ex. - Bootstrap)



- Use mobile-friendly UI controls (ex. - jQuery Mobile, Kendo UI, etc.) to make them easier to use on a mobile device (ex. - perhaps buttons instead of just hyperlinks)

DEMO: Getting Started w/ Cordova

Here we'll see:

- VS Project Templates (JavaScript and TypeScript) (i.e. - .jsproj file)
- Config.xml - Plugins
- Where to add a Connected Service (ex. - for Push Notifications)

```
<hello-world />
```

Cordova Project Structure

Solution 'BlankCordovaApp3' (1 project)

- JS BlankCordovaApp3 (Tools for Apache Cordova)
 - Dependencies
 - merges ← Platform-specific code; can override content of same name
 - plugins ← Cordova plugins that provide access to native device features
 - res ← Used for platform-specific visual assets (icons and splash screens), signing certificates, and (if needed) platform-specific configuration files
 - www
 - css
 - images
 - scripts
 - index.html ← Default home screen of the app
 - bower.json
 - build.json
 - config.xml
 - package.json
 - Project_Readme.html
 - taco.json

Your code here

Build-related files

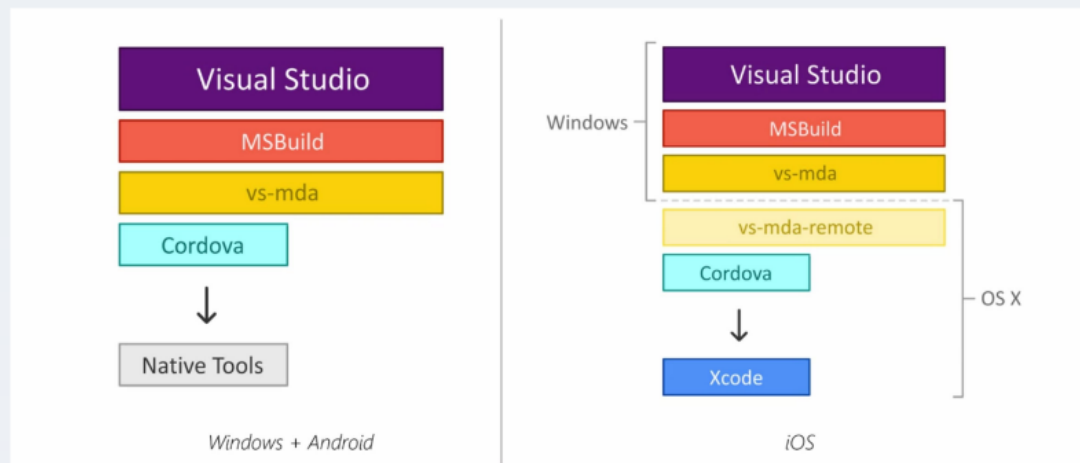
Building Cordova Apps - The 'Where'

- Building app creates a package in the debug folder
 - When you create an app using Visual Studio Tools for Apache Cordova, Visual Studio calls the Apache Cordova Command Line Interface (CLI) to build your app for the specified device or emulator.
 - The resulting packages can be accessed from the bin folder in the Visual Studio project directory.
 - APK file for Android
 - IPA file for iOS (located on the Mac)

Building Cordova Apps - The 'How'

What happens "behind the scenes" when you build your app in VS and create your final Android, Windows, or iOS package?

- MSBuild and vs-mda act as a "mediator" between what the Cordova platform expects and what your app generates
- Cordova uses the native tools to build the APK for Android, the IPA file for iOS, etc.
- For iOS, some build components are only available on the Mac, so it utilizes a Node package (vs-mda-remote). You launch the agent which "listens" for source files being passed into it by vs-mda, which in turn uses the native build tools.



Testing for iOS

How to run and debug your app on the iOS Simulator or on an iOS device ...

- Need to configure a remote build agent on Mac that "listens" for changes to your app

Install the vs-mda-remote tools:

```
sudo npm install -g vs-mda-remote --user=$USER
```

Run the agent:

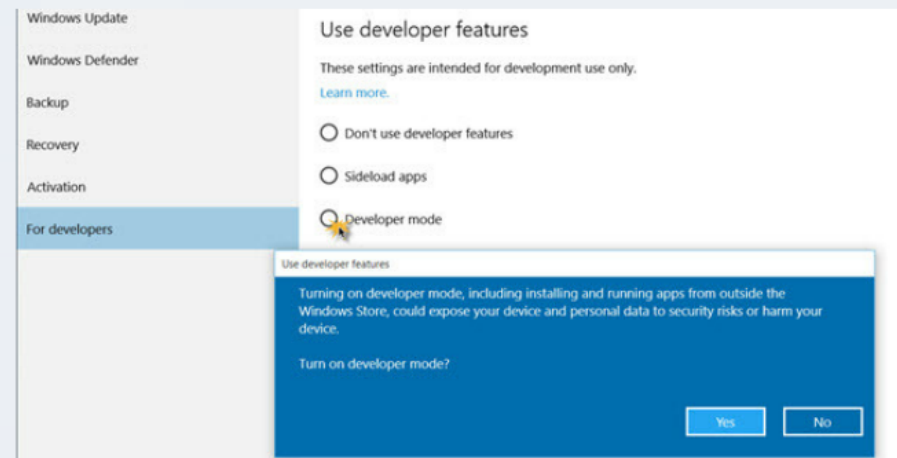
```
vs-mda-remote
```

Run the agent without added security:

```
vs-mda-remote --secure false
```

- Can also use a remote Mac cloud service if you do not want to purchase Mac hardware (ex. - MacInCloud.com)
 - You can connect to your Mac using Remote Desktop and then follow a similar setup of vs-mda-remote as noted above

Testing on Windows 10



DEMO: Debugging AngularJs Todo App

Here we'll see ...

- Sample of using a Cordova Plugin (ex. - Geolocation)
- Debugging Tools:
 - DOM Explorer
 - JS console
 - Ripple Emulator also let you do a "Live Reload Server"
 - Ripple allows for FAST dev for UI work.
 - Rebuilding and re-deploying even on fast emulators can take 10-15 sec.



DEMO: Running your app on a local web server

- The Cordova **serve** command lets you test the layout and underlying CSS of your app on your local web server. This is often helpful *before* you do device testing.
- Also great if you love some F12 developer tools (ex. - Firebug for Firefox)
- To run your app in a local web server using a native project
 - cordova platform add windows
 - cordova serve windows
 - Open a browser and navigate to the URL provided
 - NOTE: You need to rebuild the app to update its content.

Troubleshooting a Platform-specific issue

- You can build and debug an app using a **native** project
- You will find a fully generated native project under the (*\platforms*) folder in your project.
 - NOTE: These projects are deleted after every VS rebuild

Packaging Your Android App for the Store

Microsoft has a short video on YouTube covering the details

- Highlights:
 - By default, the binaries that you build for Android are **unsigned debug** builds
 - To submit an Android package (APK) to the Google Play Store, your packages need to be **signed** and built in **release** mode
 - Under the **res** folder, the **ant.properties** file specifies the location of your Google Developer key to sign the package with
 - You use Google's "Key and Certificate Management Tool" command-line utility to generate a key
 - When you update the **ant.properties** file with your new key's path and build in release mode, you will now create a **(YourAppname)-release.apk** file in the bin/Release folder that you can submit to the Google App Store

Hybrid App Alternatives

- Xamarin "Portable Razor" Development
 - Download the Portable Razor Starter Kit: <https://github.com/xamarin/PortableRazorStarterKit>
 - Uses native UI project(s) and native WebView control to launch and run the app
 - Page 'navigation' supported via intercepting the WebView's load event (i.e. - the 'before navigate to URL' event)
- Benefits
 - Core coding is in C# (i.e. - as opposed to JavaScript)
 - Features such as linq, async and await, etc. available to you
 - Familiar to ASP MVC developers
 - Once in C#, can access 100% of native APIs too, store data to a local SQLite DB, etc.

Portable Razor - Views

```
@inherits PortableRazor.ViewBase
@model System.Collections.Generic.List<Politician>

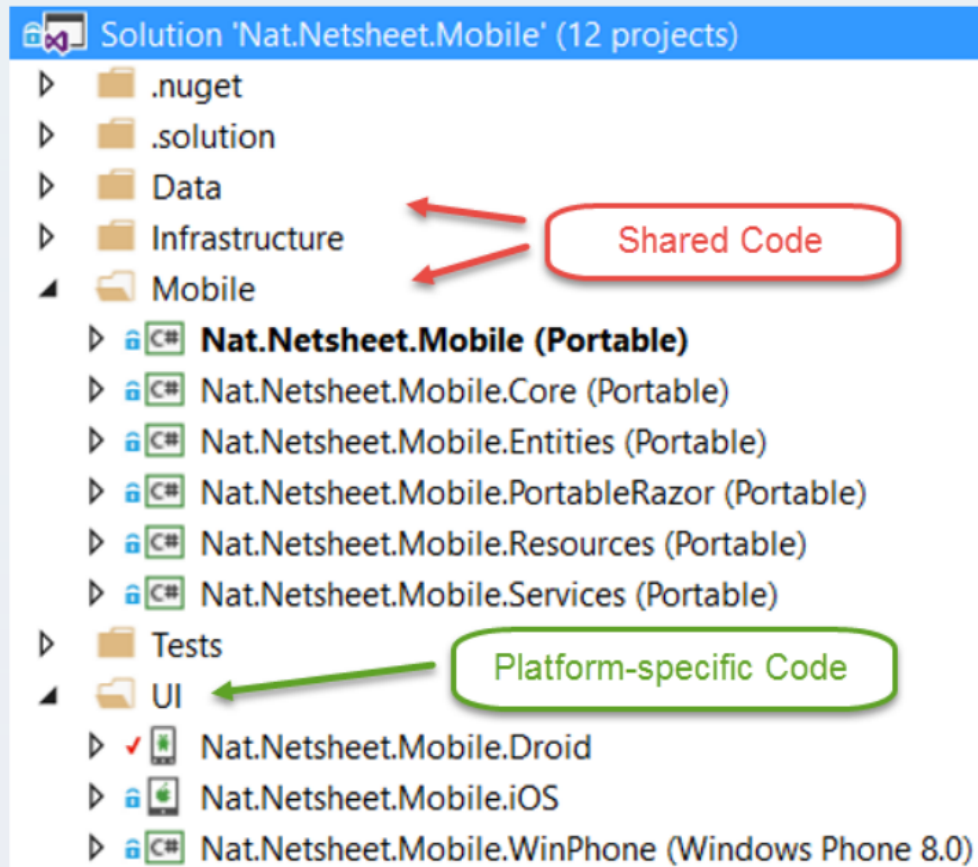
<html>
<head>
  <link rel="stylesheet" href="jquery.mobile-1.4.0.min.css" />
  <script src="jquery-1.10.2.min.js"></script>
  <script src="jquery.mobile-1.4.0.min.js"></script>
  <script src="jquery.lazyloadxt.js"></script>
</head>
<body>
  <div data-role="header" style="overflow:hidden;" data-position="fixed">
    <h1>Congress</h1>
  </div>

  <ul data-role="listview" data-inset="true" data-filter-placeholder="Search Congress..." data-
    @foreach(var p in Model) {
      <li>
        <a href="@Url.Action("ShowPoliticianView", new {id = p.Id })">
          
          <h2>@Html.Label(String.Format("{0} {1}", p.FirstName, p.LastName))</h2>
          <p>@Html.Label(String.Format("{0} {1}", p.Party, p.State))</p>
        </a>
      </li>
    }
  </ul>
  <div data-role="footer" data-id="foo1" data-position="fixed">
    <div data-role="navbar">
      <ul>
        <li><a href="@Url.Action("ShowPoliticianList")">Politicians</a></li>
        <li><a href="@Url.Action("ShowFavoriteBills")">Bills</a></li>
      </ul>
    </div>
  </div>
</body>
</html>
```

Portable Razor - Controllers

```
public Politician ShowPoliticianView(int id) {  
    var politician = dataAccess.LoadPolitician (id);  
  
    var template = new PoliticianView { Model = politician };  
    var page = template.GenerateString ();  
  
    webView.LoadHtmlString (page);  
    return politician;  
}
```

Portable Razor - Actual App



Additional Resources

Cordova 7-Part Series
YouTube Channel:



Getting Started with Apache Cordova Mobile
by Visual Studio - 1/7 videos

- Tutorial: Installing the Visual Studio Apache Cordova (1/7)
- Tutorial: Building Your First Apache Cordova App (2/7)
- Tutorial: Provisioning and Debugging Cordova Apps (3/7)
- Tutorial: Interoperability with Other Platforms (4/7)
- Tutorial: Accessing Native Device Capabilities (5/7)
- Tutorial: Building for a Specific Cordova Platform (6/7)
- Tutorial: Building for a Specific Cordova Platform (7/7)

Tutorial: Installing the Visual Studio Tools for Apache Cordova (1/7)

Visual Studio

Subscribe 20,880

5,495

Published on Apr 28, 2015

Don't spend your valuable time finding, installing, and configuring a bunch of tools and dependencies! Let Visual Studio handle it all for you, and in this "Connect, Learn!" tutorial we'll walk you through getting up and running with our free Tools for Apache Cordova.

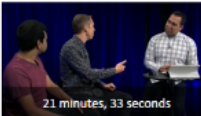
SHOW MORE

Microsoft's
Channel 9:


Search

Term:

Results



Live Q&A - Cross-Platform Mobile Development
Visual Studio 2015 Final Release Event
July 17, 2015
★★★★★ (1)
Seth talks to the Apache Cordova team.
21 minutes, 33 seconds



Building cross-platform mobile apps using C# and Visual Studio 2015
Visual Studio 2015 Final Release Event
July 17, 2015
★★★★★ (1)
Xamarin Developer Evangelist James Montemagno shows you how the Xamarin Platform enables developers to leverage their C# and .NET skills to create native mobile apps for iOS, Android, Mac, and Windows.
15 minutes, 16 seconds

What is WinJS? [Cross-Platform Development with Visual Studio: \(05\)](#)

Some Known Gotchas

- Debugging Page "On Load" Events in Android & Chrome:
 - To debug code on page load in Ripple or on Android devices/emulators: launch your app, set breakpoints, and then run "window.location.reload()" in the JavaScript Console
- DOM Explorer
May need to select the element in the HTML source instead of on the emulator
- Android Emulator Issue:

Could not create the Java Virtual Machine error: When building for Android, you may encounter a set of errors in the Errors List like the following:

```
Error    Could not create the Java Virtual Machine.
Error    A fatal exception has occurred. Program will exit.
Error    C:\cordova\BlankCordovaApp2\BlankCordovaApp2\platforms\android\cordova\build.bat: Command failed with
```

The problem is that the Ant or Gradle build systems are running out of heap memory when trying to compile your application. To resolve this problem you can increase the heap of the JVM by setting the following environment variable and restarting Visual Studio:

```
_JAVA_OPTIONS=-Xmx512M
```

See [Tips and Workarounds](#) for additional details.