Write Once, Run on Any Handset

You know, more or less...

Agenda

- Introduction
- The multi-handset approaches
  - Multiple separate native apps
  - A Web app
  - Multi-handset frameworks
    - Rhodes
    - PhoneGap
    - Appcelerator
- Summary and Q&A

Speakers

- Aaron Mulder, CTO, Chariot Solutions
- Kevin Griffin, Architect, Chariot Solutions

Overall Goal

- Build a mobile app that supports as many handsets as possible
  - Generally targeting smartphones
- Limit the time spent fooling with different platforms
- But don't sacrifice functionality

Multiple Native Apps

Best Case / Worst Case

- Let’s imagine we build a separate native app for each device
  - Using the API and SDK for each device, distribute apps in the platform’s app store or whatever
  - Using a single backend, and similar screens and architecture for each app

Best Case

- Can easily use the native look and feel for each device
- Can take advantage of many slick API features (graphics, animation, XML)
- Easy to work around handset limitations
- Really responsive, and high-performance 3D
- Great user interaction: touch, drag, shake...

Worst Case

- iPhone: Objective C
- Android: Java (custom blend of Java SE, Java ME, etc.)
- BlackBerry: Java ME plus RIM APIs
- Palm: WebOS, Mojo framework (HTML & JavaScript)
- Symbian: (C++, JavaME, .NET, Python, Qt, etc)
- Nokia: Web Runtime (WRT) -- HTML & JavaScript
- Windows Phone 7 Series: (.NET)

Verdict

- Benefits include responsiveness, and the look and feel that users are accustomed to
- Drawbacks include the huge numbers of SDKs and development platforms
  - At a minimum, need both Mac and Windows for developers!
- Mandatory to coordinate up front to share back end, app architecture and screen flow, etc.

Web App

I’m all about portable...

- ~~Most~~ Best smartphones use WebKit browsers
  - iPhone, Android, Palm WebOS, Nokia WRT, BlackBerry 5 (next)
  - HTML 5 and CSS 3
  - But, *not* Windows Phone (mobile IE)

CSS/JavaScript the key

- There are CSS and JavaScript libraries to:
  - Make your app look native on a particular platform (iUI, jQTouch)
  - Access handset features (like GPS) on a particular platform
- Plus you can use them to reformat screens for different display sizes...

But...

- At the present time can’t get a native look and feel for every platform
- Relies on a solid WiFi/3G connection, and still not as performant
- Detecting the specific device is up to you (with the help of WURFL, MobileFu, etc.)

The Good and the Bad

- Some success stories
  - GMail on mobile
  - iSepta
  - Did you try phillyemergingtech.com ?
- Some sites with issues
  - Links in break when redirected to mobile site
  - Broken flipping between mobile and ‘full’ site

Verdict

- Only basic Web technologies needed, and they have decent functionality
- Can use any back-end technology you want (Java, Ruby, .NET, whatever)
- In the end, may still need customizations for each style of handset (screen size, etc.)
- And separate "mobile" Web sites can have their own usability issues

Multiple-Handset Frameworks

Common Elements

- You write code in some single environment
- A project is generated for each platform and needs to be built/run
  - With varying degrees of automation
- You get a native app, with some framework code and some of your code
- Debugging options are limited

Differentiators

- What language(s) do you code in?
- How many platforms are well-supported?
- What native features are supported?
- What’s the look and feel?
- How easy is the development process?
- Is there any cost?

RhoMobile

Highlights

- 3 Components
  - Rhodes -- multi-handset development
  - RhoSync -- integration
  - RhoHub -- hosted IDE/build system
- Commercial with varying costs (except for open source mobile apps)

Rhodes Technology

- Write your app logic in Ruby
- Includes MVC Web framework and an ORM layer
- Screens in HTML & JavaScript
- Packages a Ruby interpreter with your app on the phone

Extras

- RhoSync
  - Easily pull information from existing back end sources into a Rhodes app
  - Write adapter logic in Ruby
- RhoHub
  - Online development/build environment (need no SDK on local machine)
  - Hosted RhoSync adapters

Build Process

- Run command-line scripts to create project
  - Creates an XCode project for iPhone
  - Creates an APK for Android
  - Creates Visual Studio project WM.
  - Creates a (Windows) SDK proj. for Blackberry
- Then build that project like a native one

Result

- Typically not the native look and feel
    - Though you can create some elements like a tab bar with the Rhodes API
    - And you can use libraries like iUI
- Can access a variety of native features depending on the platform
- Best for iPhone, Android and Blackberry (Windows Phone 7 under discussion)

Verdict

- Great if you have Ruby skills
- Only framework with integrated Web-MVC and ORM option
- Great if you want easy sync to existing apps or data stores
- RhoHub can be nice to develop for multiple phones from one machine
- But not free to develop non-OSS apps

PhoneGap

Highlights

- Build apps using only HTML & JavaScript, yet still get the native app experience
- Excellent build scripts to create the native projects
  - iPhone, Android, Palm, BlackBerry, plus Windows Phone, Symbian
- Exposes native phone features via a JavaScript API

Technology

- All the power of HTML5/CSS3
  - Plus SVG for fancy graphics?
- Prefers XUI (like JQuery), but what the heck, you could put any JS stuff in there
  - Best to use JSON for talking to a server
- Runs locally on phone -- performs OK and no issues with cross-domain JavaScript

Disadvantages

- Uses phone's features for a database, etc.
    - Not really standard beyond 'local storage'
- All your validation/logic/etc. is in JavaScript... hopefully not embedded in every page (see 'before Web MVC')
- Debugging options are pretty limited
    - Maybe run in Firebug, if not using native phone features

Result

- Minimal funny skills required to develop for many phone platforms (HTML & JS)
  - But still need to drive XCode, Visual Studio, Java compiler, etc. -- especially if you want to extend the PhoneGap API
- Similar look and feel issues as regular Web apps
- Great for something that could be a regular Web app, but would be just a little bit better with native features & app store
- Great handset support (and lowest cost) of all these frameworks

Titanium

Highlights

- Also uses HTML, CSS, & JavaScript
- But has a JavaScript API for native widgets to get a native look and feel
- Great tooling builds native executables under the covers
- Limited platform support (iPhone, Android, and BlackBerry coming) (iPad just added!)

Technology & Issues

- Pretty much the same starting point as PhoneGap (including limited debugging)
- Uses “Titanium” runtime (also a desktop version available)
  - Can’t run in a browser
  - Titanium API has some nice features that give you a few more options

Results

- More native look and feel, despite using only Web technologies
- Very easy development process (e.g. click "Run in iPhone Simulator" button)
  - Don't even need to know XCode
- Still the native app experience
  - All Web stuff runs locally, like PhoneGap

Verdict

- Best multi-platform development experience
  - If you're OK with the cost and the limited number of platforms supported
- Nice to have the native UI option
- Still best for Web 2.0 ninjas
  - If JavaScript isn't your thing, then...?

Wrap-up

Decisions, Decisions

- If no native features, maybe a Web app is the way to go (esp. with HTML5 features)
  - Or PhoneGap if you want the app store experience
- If maximum performance or native features are key, maybe you need multiple native apps
- Titanium is easiest framework to use, but with fewer supported platforms

Architecture Issues

- Do you want code in Ruby, HTML/CSS/JS, or the native languages?
- Do you want code living on the handset or on a Web server?
  - Remember: old releases may never die
- Do you favor an MVC Web framework?
- Does your back end support JSON?

Q&A

ammulder@chariotsolutions.com
kgriffin@chariotsolutions.com