PhoneGap Development Best Practices

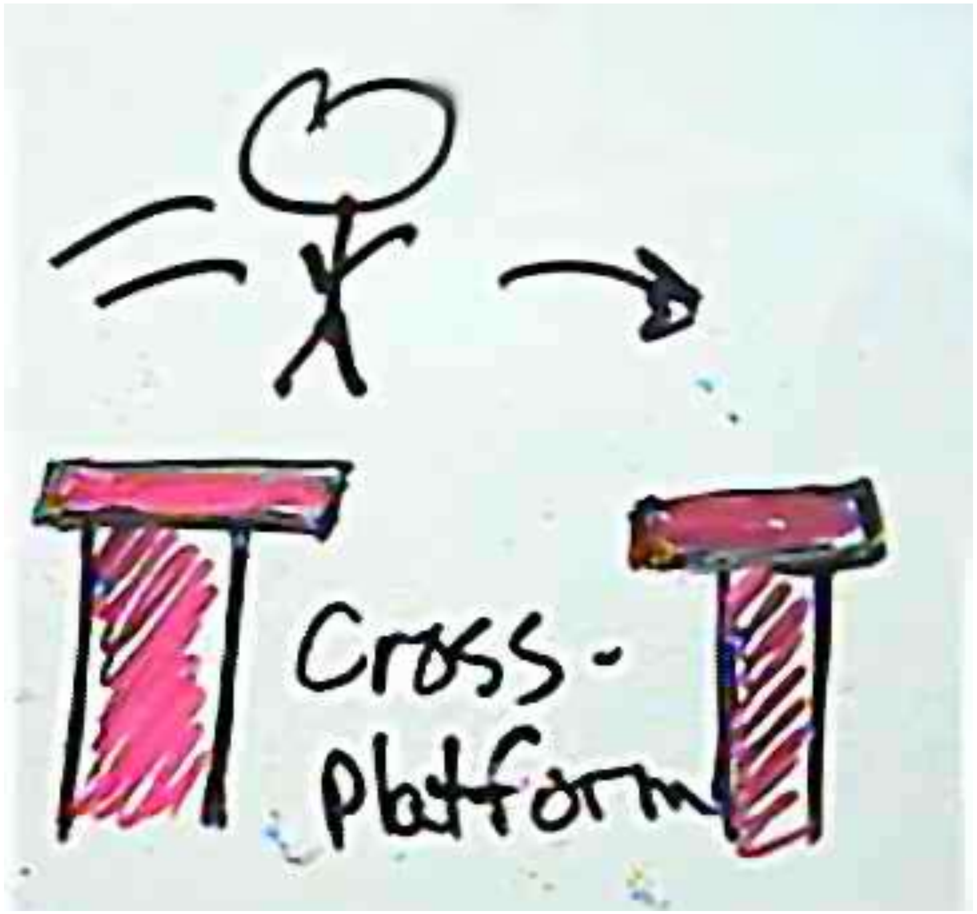
Fil Maj
Nitobi Software

Overview

- Fast cross-platform prototyping, how to do it.
- Templating.
- Offline strategies.
- Persistent storage.
- File API.
- XUI overview.
- When *not* to use PhoneGap.

Cross-platform Prototyping with PhoneGap

- Main advantage of PhoneGap is you can reuse your web application source code across platforms.
- A good, quick approach is to write one set of assets and 'tweak' across platforms.
- HTML and CSS can be, more-or-less, written once.
  - Highly recommend writing percentage-based styles initially, and down the road tweaking on a per-platform basis.
  - Combined with a <meta name="viewport"> tag (which tells the browser the size of the screen it's being viewed in), results are good.



Cross-platform Prototyping with PhoneGap, cont.

- JavaScript is tougher, mainly due to weak BlackBerry support. However, latest XUI works on BlackBerry! More on this in a bit.
  - A process that has brought a measure of success for us is to first determine which platforms to target, and then build your mobile application logic in a JavaScript framework that works with the ‘lowest common denominator’ platform. JavaScript frameworks are your friends (jQuery, MooTools, XUI, etc.).
  - Existing software development best practices apply. Use a layered and modular approach for your JavaScript.
  - MVC (model-view-controller) paradigm is great for prototyping, as you can revisit and, if need be, recode particular modules of your app as you iterate.
    - Model = PhoneGap JS API + offline storage/cache (+ Lawnchair?)
    - Controller = JavaScript
    - View = HTML + CSS

Templating

- Common web development practice used to encapsulate view components of an app.
- As you work on an app, you notice repeatable HTML/CSS patterns that come up. Don’t copy+paste it!
  - Encapsulate the view pattern HTML aka ‘template’ in a JavaScript string:
    ```
    var tweetTemplate = ‘<div class=“tweet”>{USER} said {MESSAGE}</div>’;
    ```
  - Create a JavaScript function that takes the template and replaces the tokens (placeholders) with actual data:
    ```
    function tweetify(data) {
      var tweets = document.createElement(‘div’);
      var newHTML = ‘’;
      for (var i =0; i<data.length;i++) {
          var tweet = tweetTemplate.replace(/{USER}/,data[i].user);
          tweet = j.replace(/{MESSAGE}/,data[i].message);
          newHTML += tweet;
      }
      tweets.innerHTML += tweet;
      document.getElementById(‘content’).appendChild(tweets);
    }
    ```
- This approach follows the MVC paradigm and is a good example of separation of concerns. Also easy to read, easy to extend and reusable.
- John Resig blogged about this, he doesn’t use regex but instead he uses: `tweetTemplate.split(“match”).join(“replace”).`
  It’s quicker apparently: http://ejohn.org/blog/javascript-micro-templating/

Offline Strategies

- Inherently, mobile devices will not be networked all the time. Bad coverage, on the plane, no data plan, etc.
- Extremely important for every application to take this into account, especially for iPhone. It is a hard criteria in Apple's App review process (to see how gracefully an app handles lack of internet connection).
- PhoneGap offers reachability API, example: http://github.com/phonegap/mobile-spec/blob/master/tests/network.tests.js

Persistent Storage

- iPhone has SQLite, Safari implements the HTML 5 spec. Tutorial: http://phonegap.pbworks.com/Adding-SQL-Database-support-to-your-iPhone-App
  - Note: each page in a PhoneGap app can have only a single database object open, and its maximum store size is 5 megabytes (2 MB on Android, but you can change this to suit your needs). Take this into account when using a single-page approach to a PhoneGap app (more on this later).
- BlackBerry has ‘persistent storage’ – a giant key/value store. Code exists to access this, but we haven’t figured out how to fit it into the PhoneGap API. Somehow it needs to align with the other platforms. Ask me (filip.maj@nitobi.com) for it if you want it!

File API

- You can read/write files from PhoneGap too. Implementations available at our own local repositories (not yet part of the spec / public API).
  - http://github.com/purplecabbage/phonegap-iphone/blob/master/PhoneGapLib/javascripts/core/file.js
  - BlackBerry native code is present but needs a JavaScript wrapper.
  - Android File I/O got revamped recently.
  - Also: Mobile Spec tests are NOT up to date and platforms are fragmented on the implementation of this feature. It needs some work!
- Work is ongoing to line our file API up with the HTML 5 specification: http://www.w3.org/TR/2009/WD-FileAPI-20091117/#dfn-empty

XUI Overview

- Another (!) JavaScript framework: xuijs.com
- This one's special, though: specifically designed for mobile devices.
- Biggest win: small footprint (~6-10kb).
- Inspired by jQuery with a very similar syntax.
- Works on iPhone, Android, Symbian, Palm and… drum roll… BlackBerry too! But you need to build it specifically for BlackBerry; the output script files are labelled as core, more or bb – the bb file is for BlackBerry.
- Is the recommended framework to use with PhoneGap.

PhoneGap Performance Tips

- For small apps, use a single HTML page.
  - Use JavaScript to show/hide page elements based on user interaction instead of linking to a separate page.
  - Especially important for BlackBerry, since each new page request forces the device to encode requested assets into base64 on-the-fly.
- Obfuscate / crunch your JavaScript before release.
  - Devices only reserve a bit of memory for JavaScript interpreters for the browser. If your JavaScript is small enough, it won’t be necessary for the browser to constantly page your scripts in/out of the browser memory. iPhone, for example, has 25kb of memory reserved for JavaScript parsing.

PhoneGap Limitations

- As important as knowing how to use PhoneGap is also knowing when NOT to use it.
  - Complex games, intensive graphics. Use OpenGL for that, not PhoneGap.
  - For slower phones (not iPhone, not Nexus One, not Xperia X10), PhoneGap apps using the latest interactive Google Maps APIs tend to be slow. Static maps OK, though.

Questions/Comments?