Modern Web Development Part 6:
Apache Cordova

Overview

In the Bad Old Days (™), Web development consisted of HTML, Javascript and a plain text editor. The technology has now matured to the point where we can put together a full, end-to-end development workflow that includes dependency management, build automation and debugging.

Modern web development is not about having one, all-purpose software application that tries to do everything. It's about putting together a kit of smaller, single-purpose tools and connecting them together in a way that fits your coding style.

This guide looks at a popular open source framework for deploying mobile web applications, Apache Cordova. Cordova adds a native code "wrapper" to your web app, allowing it to access mobile device hardware like the GPS or the accelerometer. You could write your mobile app in its native language (ex IOS - Objective C/Swift, Android - Java) but with a tool like Cordova you can write a single Web app and deploy it on multiple mobile platforms.

You Have:

- MWD Toolkit (see Part 1 for details)

Installation

The Cordova command line tools run on Node.js so we can install them with npm. There are additional dependencies to install based on what mobile platforms you are supporting. Refer to the Cordova platform guide (https://cordova.apache.org/docs/en/latest/guide/platforms/) for more information.

For a basic installation, open a command prompt and run the command:

```bash
npm install -g cordova
```

To get a complete list of command line options, type cordova by itself.
Usage

You can use the cordova command all by itself to set up your project. There are also several Yeoman generators that will scaffold your project. For now we'll just use cordova.

First we need to install any SDKs we may need for the platforms we are targeting. For example:


(See the Cordova Platform guide (https://cordova.apache.org/docs/en/latest/guide/platforms/) for more information.)

The simplest way to create a Cordova project is by running the command (assuming your project folder is called MyApp):

cordova create MyApp

Now we need to add the platforms we plan to support (ex. IOS). Run the following command inside of your project folder:

cordova platform add ios

To find out what platforms are already installed, run:

cordova platform ls

Let's examine the contents of our project folder:

<table>
<thead>
<tr>
<th>File/Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hooks</td>
<td>Contains scripts added by application or plugin developers to customize cordova commands.</td>
</tr>
<tr>
<td>platform</td>
<td>Contains build information for each platform to which you are deploying your app</td>
</tr>
<tr>
<td>plugins</td>
<td>Contains code to access device-specific functionality</td>
</tr>
<tr>
<td>www</td>
<td>This is where your application code resides</td>
</tr>
</tbody>
</table>
**config.xml**

A global configuration file for your app, written in XML format according to the W3C Packaged Web Apps specification (http://www.w3.org/TR/widgets/)

**NOTE:** Do NOT edit any files in the *platforms* folder. These are managed by the *cordova* utility.

Let's look at the contents of our index.html file, specifically the body:

```
<body>
  <div class="app">
    <h1>A p a c h e   C o r d o v a</h1>
    <div id="deviceready" class="blink">
      <p class="event listening">Connecting to Device</p>
      <p class="event received">Device is Ready</p>
    </div>
  </div>
  <script type="text/javascript" src="cordova.js"></script>
  <script type="text/javascript" src="js/index.js"></script>
</body>
```

This is the Cordova equivalent of "Hello World". When your app starts up, you will see the text "Connecting to Device" followed by "Device is Ready".

Now how do we see this in action? First we need to build our app (compile the code and package it in the proper device format).

In our project folder, run the command:

cordova build

This will run a build for all installed platforms. If you only want to build for a specific device (ex IOS), run:

cordova build ios

This build command is actually a combination of two cordova commands:

cordova prepare ios
If you want to use the default IDE for the platform (in our case, XCode) to modify and compile your platform-specific code, just run:

cordova prepare ios

There will be an IDE-compatible project folder under the appropriate platforms subdirectory. Just open this in your IDE and work with it like you would any other project.

Now how do we test our app? We have two options: using an emulator or directly testing it on our device.

To use the emulator, make sure that you have the appropriate SDK for your platform installed and run the command:

cordova emulate ios

(Replace ios with whatever installed platform you're currently testing.)

You'll see several screens of messages, as each build step completes:

```
** BUILD SUCCEEDED **

No target specified for emulator. Deploying to iPhone-6 simulator
io.cordova.hellocordova: 8257
** RUN SUCCEEDED **
[~/.MyApp] ruby-2.2.1 $  
```
Then the emulator will start up and load your app:

![Emulator Screen](image)

Depending on what platform emulator you are using, you should be able to simulate many device functions (Home, Lock, Reboot, Rotate Left, etc.) from the emulator menu. You may even be able to switch the emulated device as well. For example, when I switched from iPhone 6 to iPad Air:
The other way to test your app is to actually run it on a physical device. You will have to set up your mobile device for testing. This procedure differs by platform, so check the Cordova platform guides (https://cordova.apache.org/docs/en/latest/guide/platforms/index.html) for your particular device instructions.
If you have an appropriate device setup for testing, plug it into your computer and run the command:

cordova run ios

(As before, replace ios with the installed platform you are testing.)

That's enough to get you started with Cordova. See the documentation (https://cordova.apache.org/docs/en/latest/guide/overview/) for more details.