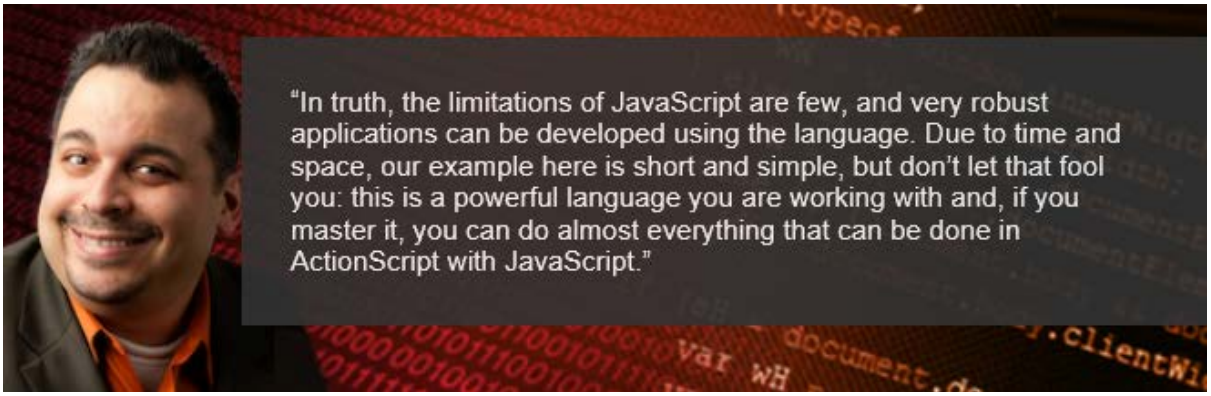
Creating an HTML5 Mobile App with PhoneGap (Jul 12)

By Mark Lassoff
July 19, 2012



It's a mobile world. We just have to live in it.

With almost universal agreement that (at least in part) the future of eLearning is mobile, mobile learning practitioners have been seeking and creating mobile tools. While many of the efforts have been laudable, many are stuck in the dated Flash-in-frame paradigm that existing tools and SCORM compliance have forced our industry into.

"Will it run on iPad?" is the question of the year—and unfortunately, much more often than not, the answer is "No."

However, there is a solution. This solution will run on iPad, iPhone, Android, and even desktop-based machines. It does, however, require thinking about eLearning production in a different way. Yes, it requires us to write some code – but that is a small price to pay for creating truly cross-platform solutions that you can create once and deliver everywhere. Perhaps upon adopting this HTML5 based standard we can drop the "e" or "m" prepended and just call what we do "learning."

What you need

This is one of those times it really helps to have a Mac. If you are seeking to create mLearning apps for iOS, unfortunately the tools are only available on a Mac. If you have a PC, you can create applications targeted to Android; however, a Mac lets you create both iOS and Android applications. (Maybe this is your excuse to finally ask the boss for a new MacBook Pro?)

You will need to download the software development kits for iOS or Android or both. For iOS development it's very easy: simply download the latest version of Xcode from the App Store. If you want to create Android applications, the process is a bit longer and more arduous – but certainly doable. Go to http://developer.android.com/sdk/index.html for detailed instructions.
Once you have either Xcode or Android's Eclipse environment installed, it's time to download PhoneGap (now officially called Cordova). Go to http://phonegap.com/download and download the latest version (see Figure 1).

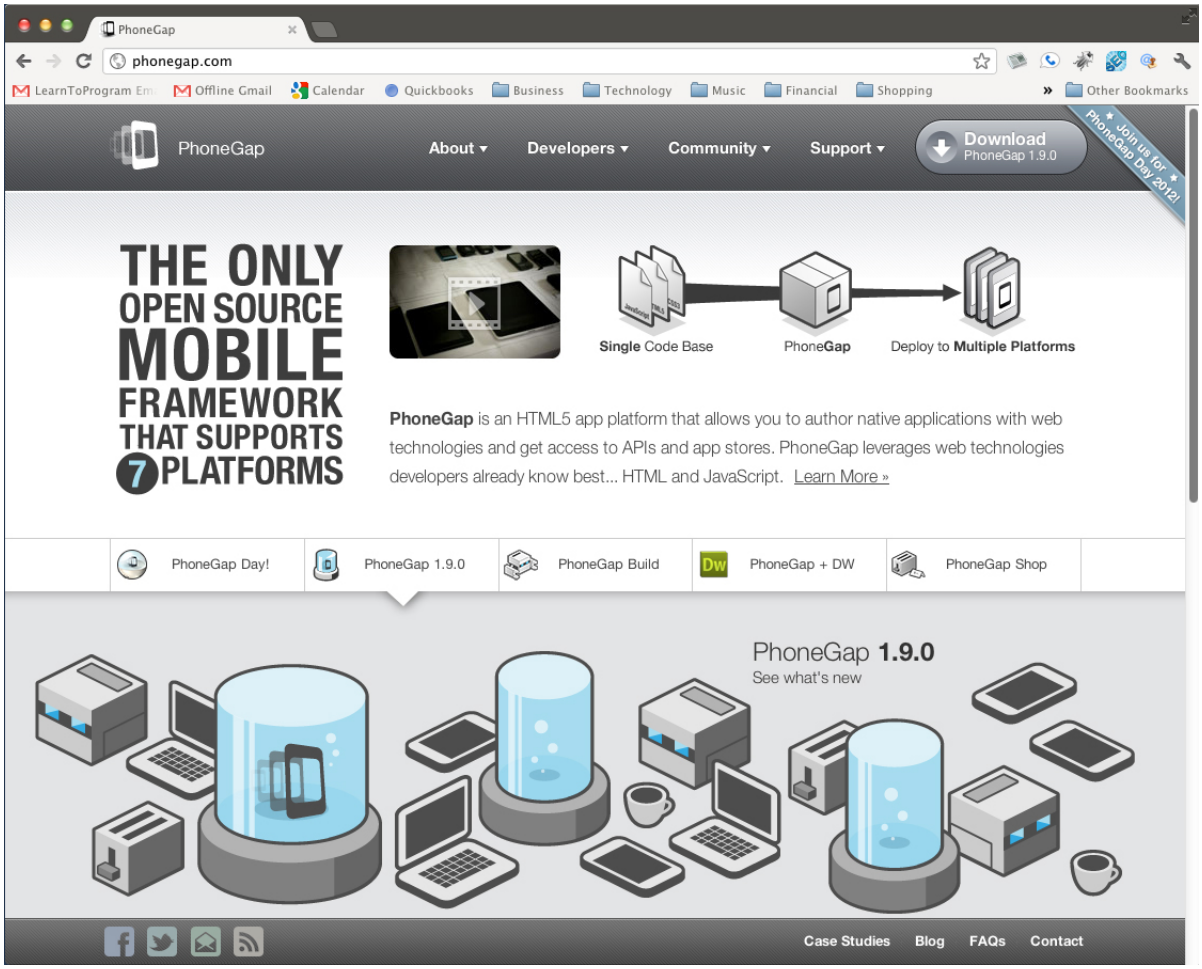


Figure 1: **PhoneGap has a very informative web presence, with complete API documentation available**

For this tutorial I am going to be using the Mac version to produce an iOS application. You can apply the information in this article to create an Android version of the same application by carefully following the "Getting Started" instructions at http://docs.phonegap.com/en/1.9.0/guide_getting-started_android_index.md.html#Getting%20Started%20with%20Android.

For the Mac iOS version, you'll have to do a very brief installation and then you'll be ready to get going! See http://docs.phonegap.com/en/1.9.0/guide_getting-started_ios_index.md.html#Getting%20Started%20with%20iOS for detailed instructions on getting PhoneGap started on the Mac.

Creating your application

When you create your application, PhoneGap essentially builds a native application to display your HTML5 application in a browser control. In addition, PhoneGap builds bridges between the hardware and your application, which allow you to code the hardware in JavaScript instead of the native Java (for Android) or Objective C (for iOS). You won't, however, have to touch these native parts of the application very much at all. You'll use familiar HTML, CSS, and JavaScript to create your application. In Xcode you can create a new project by selecting the appropriate option from the file menu. From the next screen you can begin configuring your application by selecting the Cordova template (Figure 2).

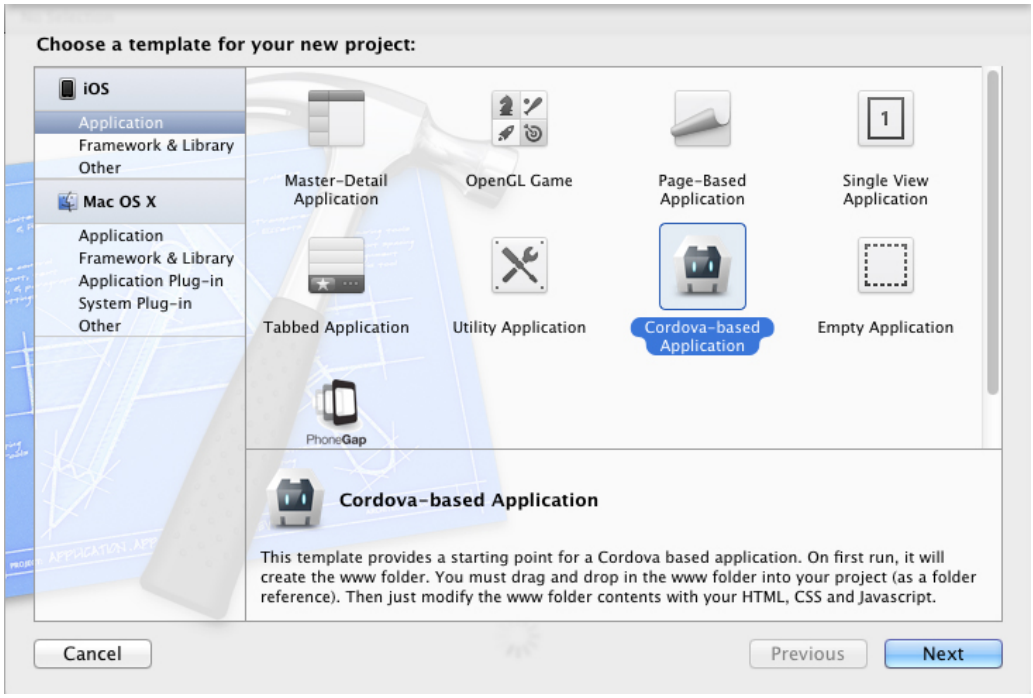


***Figure 2*: Selecting the Cordova template in Xcode**

Click Next and on the next screen you will need to provide an application name (called product name in Xcode) and a Company Identifier. The application name can be whatever arbitrary name you'd like to give your app. The company identifier is used to uniquely identify your app in the Apple Store and distinguish it from other applications on the device. I recommend that you use a backwards version of your Web URL, as in Figure 3.

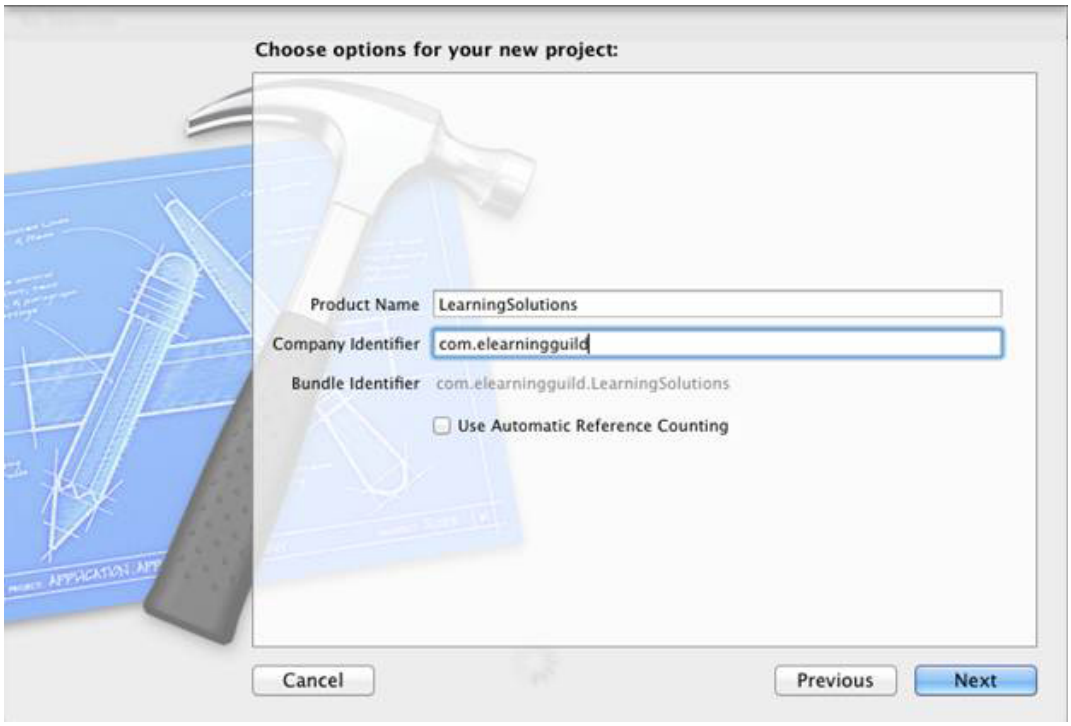


***Figure 3*: For the company identifier field use a backwards version of your domain name – so if your domain is educationalmusic.com, use com.educationalmusic**

Make sure the check is removed from the automatic reference counting option. While this is a useful option when making native iOS applications, keeping this option checked will cause your PhoneGap applications to break. Click Next, and on the following screen you'll choose a location to save your project files.

The next step is to run the PhoneGap application. Press the Run button in the upper left hand corner of the Xcode interface.

In a few moments the iPhone or iPad emulator will become visible and you will see an error message on the screen (Figure 4). Don't worry! The error message is exactly what we want to see at this point.

Figure 4: **The iPhone/iPod/iPad emulator displaying the error we are looking for**

In addition to displaying an error, PhoneGap created a new folder in your project. That folder is called "www". Navigate to your project folder in your file system and drag the newly-created www folder into Xcode. You want to make sure you drag the www folder directly on top of your project name in the upper-left-hand corner of the Project Navigator panel on the left side of the screen. For the options dialog box that appears next, select your options as Figure 5 shows. Figure 6 shows the results.

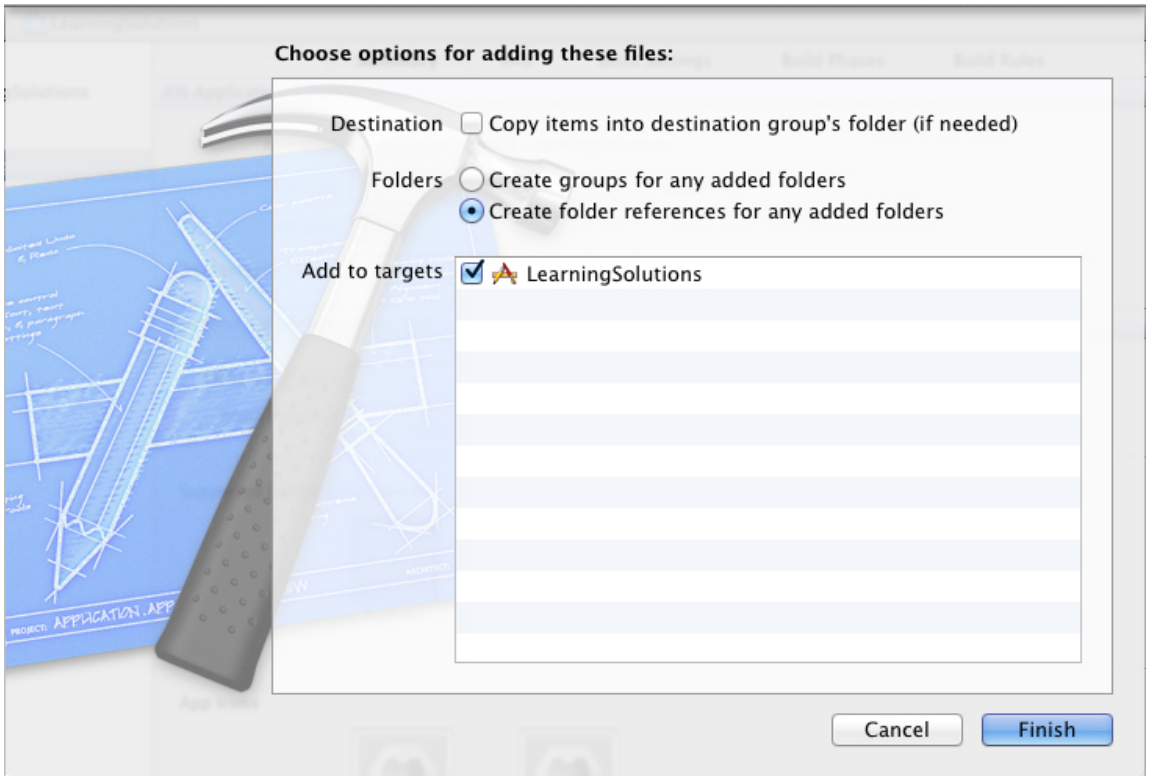


Figure 5: **Correctly configured file options dialog box**

Figure 6: **If you've done everything correctly, your Project Navigator on the left side of the screen should appear like this**

At this point, you should run your project once more. If you have been successful, a message that says "Hey, it's Cordova!" should appear, as well as an Alert Dialog box which you will have to acknowledge (Figure 7).

Now, return to Xcode. Open the file called index.html in your new www folder. This is the starting point for creating your application. Congratulations!

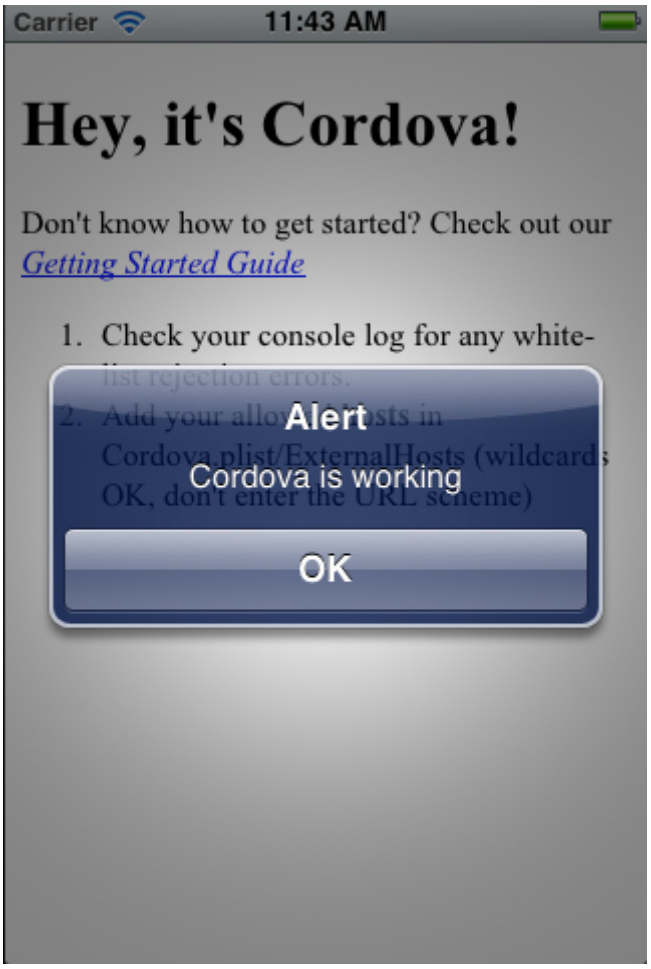


***Figure 7*: The default Cordova greeting screen viewed on the iPhone simulator**

Creating an informational page

On the index.html page, scroll down to the bottom part of the code and you'll see what looks like (because it is!) everyday HTML code. This is the code that Cordova is processing and placing in the browser to become your app. If you can code basic HTML, you can change what appears here.

Let's erase the templated information so we can insert our own code and content. This is a short two-step process. First, erase everything between the opening and closing body tags. Once you do so you should have something like this in your code:

```
view plain  copy to clipboard  print  ?
01.  </head>
02.    <body onload="onBodyLoad()">
03.    </body>
04.  </html>
```

The second step is to eliminate the alert pop-up that appears when the application loads. To do so locate the `onDeviceReady()` function. It should appear directly above the area in which we just erased the body tag content. Inside that function you'll find a like that reads:
`navigator.notification.alert("Cordova is working");`

Immediately preceding navigator place two forward slashes (//) – this will convert that line into a comment that will be ignored by the device.

We're going to add some code that you might find in an mLearning application designed to teach HTML. Place your cursor after the opening body tagline. Place the following code there:

```
view plain  copy to clipboard  print  ?
01. <body onload="onBodyLoad()">
02.   <h1>What is HTML?</h1>
03.   <p>HTML stands for HyperText Markup Language. It is the language used to
04.        define the purpose of elements that appear on a web page. HTML is
05.        interpreted by browsers (both mobile and web browsers) and displayed
06.        according the rules in its associated style sheet.</p>
07.   <h3>Sample HTML</h3>
08.   <img src="http://50.63.156.60/img/note16.gif" />
09.     <div id="question">
10.       <h3>Question for Understanding</h3>
11.       <p>What is the purpose of HTML?</p>
12.       <input type="button" value="Adjust the background color of Pages"/>
13.       <br/>
14.       <input type="button" value="Define the purpose of page elements"/>
15.       <br/>
16.       <input type="button" value="To communicate with the server"/>
17.       <br/>
18.       <input type="button" value="Determine how a page will be styled"/>
19.     </div>
20. </body>
```

You may run your app at this point. You'll notice that the link to the image is broken – which is to be expected at this point because we have to take one more step.

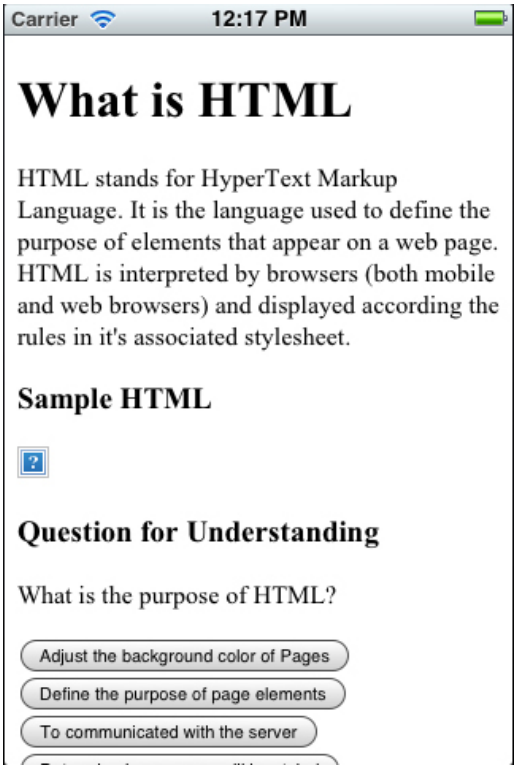


Figure 8: **Content as the iPhone simulator displays it**

In your project navigator, open the folder called “Supporting files” and select the file named “Cordova.plist”. This is a list of properties relevant to PhoneGap. Click the arrow next to ExternalHosts and then the “+” sign that appears. In the space available, type an *. This will whitelist any website you want to access – including the one that holds the image we’re trying to display.

Note: Sometimes the iOS simulator has difficulty displaying images from an external URL. Don’t worry too much if it appears as a broken link.

Adding some interactivity

Add a bit of JavaScript and you can easily add interactivity to your application. If you’ve worked with Flash ActionScript 3.0 in the past, JavaScript is closely related – it’s based on the same ECMA standard, and has an identical syntax. We’re simply going to add some JavaScript to our `<input>` tags to give the user some feedback as to whether their answer is correct or not.

In truth, the limitations of JavaScript are few, and very robust applications can be developed using the language. Due to time and space, our example here is short and simple, but don’t let that fool you: this is a powerful language you are working with and, if you master it, you can do almost everything that can be done in ActionScript with JavaScript.

Let’s change the section with our input tags as follows:

```
view plain  copy to clipboard  print  ?
01.  <input type="button" value="Adjust the background color of Pages"
02.    onclick="navigator.notification.alert('Sorry, incorrect');"/>
03.  <br/>
04.  <input type="button" value="Define the purpose of page elements"
05.    onclick="navigator.notification.alert('Correct Answer!  Good work!');"/>
06.  <br/>
07.  <input type="button" value="To communicate with the server"
08.    onclick="navigator.notification.alert('Sorry, incorrect');"/>
09.  <br/>
10.  <input type="button" value="Determine how a page will be styled"
11.    onclick="navigator.notification.alert('Sorry, incorrect');"/>
```

Run your application and you’ll notice the buttons will now react to clicks, displaying a dialog with feedback for the user (Figure 9).

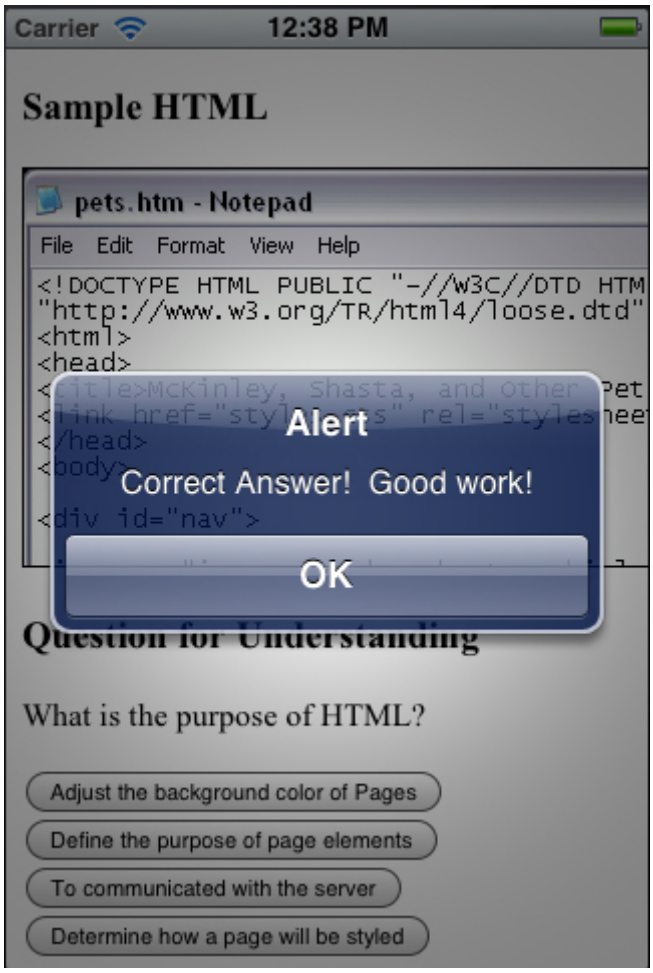


Figure 9: **The complete application operating after the user provided the correct answer**

Congratulations … you have completed your first mobile app! Have a sandwich and the beverage of your choice!