# Native Code Apps versus HTML5 Apps

# What are Native Code apps?

- Apps that you build using XCode, Objective-C and Interface builder
- The user interface uses entirely or mainly UIKit
- The user interface uses Storyboards and XIBs
- The app can *directly* access all of the hardware features through the system APIs.
- The app can take advantage of iOS's native memory management and Foundation classes

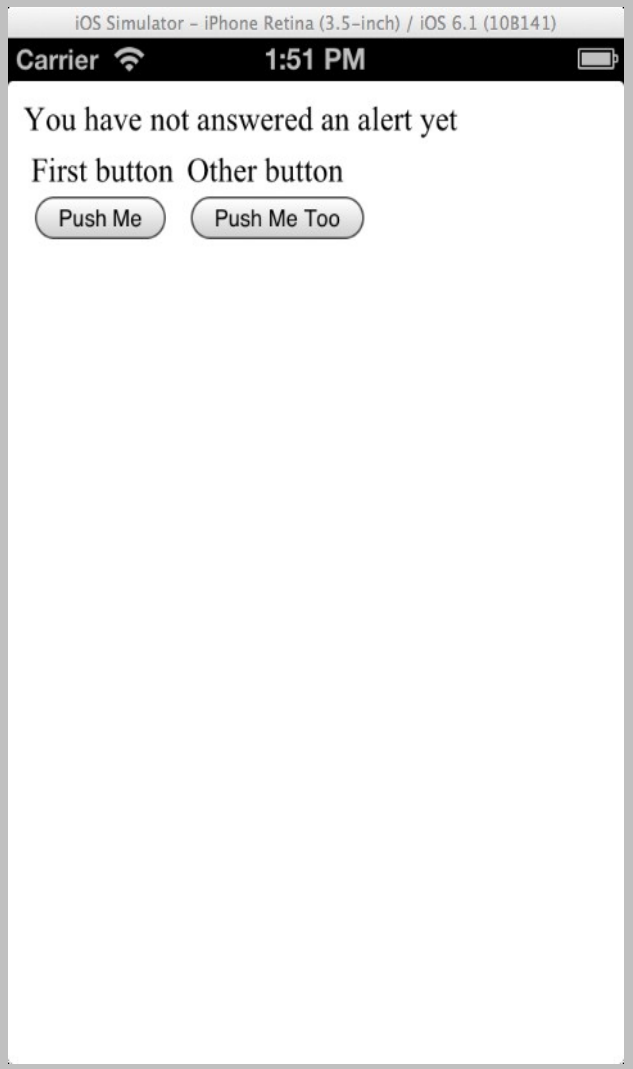
# What are HTML5 apps?

- Apps where the user interface and app logic is implemented in a combination of HTML, CSS and Javascript.
- Pure HTML5 apps (also known as “web clips” in iOS) only use HTML, CSS and Javascript (plus images and other binary assets).

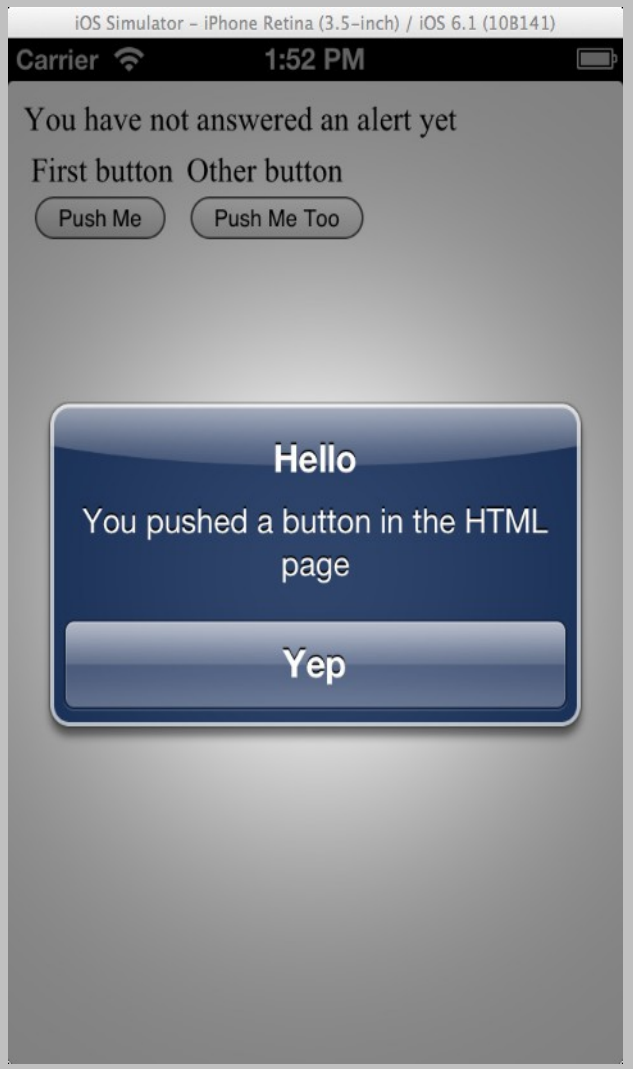
# What are Hybrid apps?

- Hybrid apps are just that – the UI and varying amounts of the app logic are written in HTML / CSS / Javascript but have a native app shell wrapped around them.  (The native app shell males access to features like the camera or the GPS hardware available to the web UI.)
- The app shell can be homegrown or it can be open source (e.g. Phonegap).

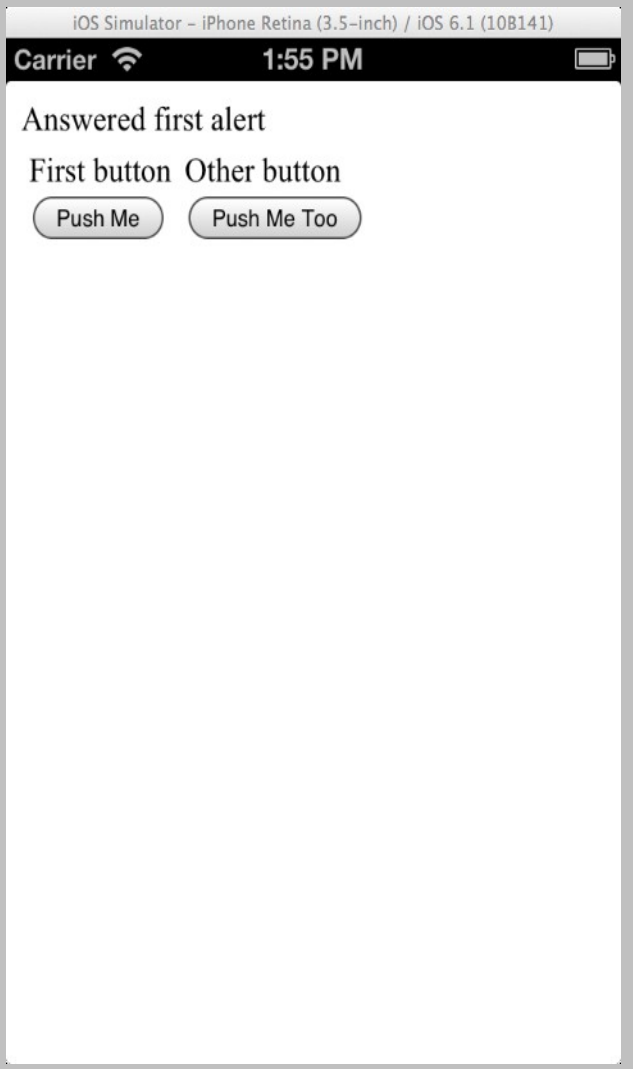
# Sample HTML UI




Tap 'Push Me' in HTML




Native code displays UIAlertView




Native code updates HTML

# HTML UI file for sample hybrid app

```
<html>
<head>
<meta name="viewport" content="width=device-width;
   initial-scale=1.0; maximum-scale=1.0; user-scalable=0;"/></head>
<body>
<div id='labelDiv'>
You have not answered an alert yet
</div>
<table border=0 cellpadding=2 cellspacing=2>
<tr>
<td>
First button<br>
<button onclick="document.location='launchHelloAlert'">Push Me</button>
</td>
<td>
Other button<br>
<button onclick="document.location='launchYouAgainAlert'">Push Me Too</button>
</td>
</tr>
</table>
</body>
</html>
```

# How HTML talks to native code

How do widgets in the HTML communicate with the native code? Through a URLWebViewDelegate method in the ViewController. Each time the UIWebView tries to load a URL, this method is called and the URL is supplied as a parameter. The return value from this delegate method lets the UIWebView know whether to try and load the URL or not.

```
-(BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:(NSURLRequest *)request navigationType:(UIWebViewNavigationType)navigationType
{
    BOOL tryAndLoadURL = YES;
    NSString *urlString = [[request URL] lastPathComponent];

    if([@"launchHelloAlert" isEqualToString:urlString])
    {
        UIAlertView *helloAlert = [[UIAlertView alloc] initWithTitle:@"Hello" message:@"You pushed a button in the HTML page" delegate:self cancelButtonTitle:@"Yep"
otherButtonTitles:nil];
        [helloAlert show];
        tryAndLoadURL = NO;
    }
    else
    if([@"launchYouAgainAlert" isEqualToString:urlString])
    {
        UIAlertView *helloAlert = [[UIAlertView alloc] initWithTitle:@"Hello2" message:@"You pushed the other button in the HTML page" delegate:self cancelButtonTitle:@"Yep"
otherButtonTitles:nil];
        [helloAlert show];
        tryAndLoadURL = NO;
    }

    return tryAndLoadURL;
}
```

# How native code talks to HTML

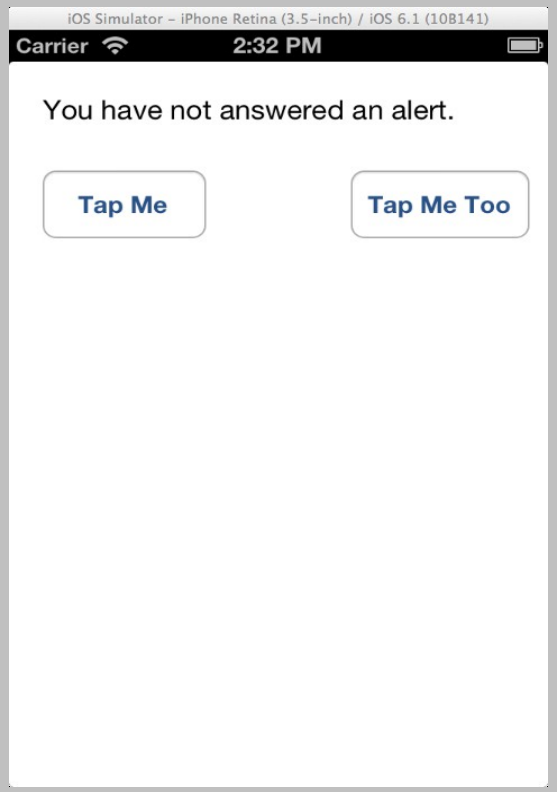
If the native code needs to change something in the HTML there are two ways to do it: reload the HTML (awkward) or execute Javascript in the HTML (less awkward).

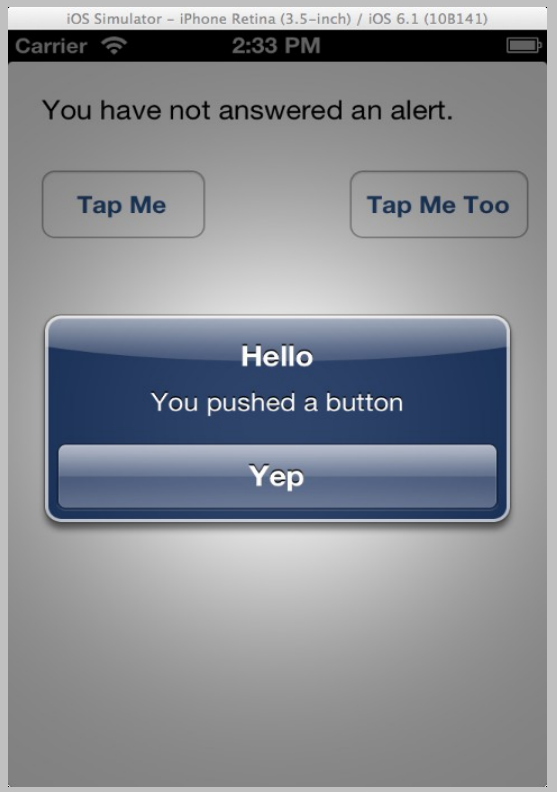
```
-(void)alertView:(UIAlertView *)alertView didDismissWithButtonIndex:(NSInteger)buttonIndex
{
    if([@"Hello" isEqualToString:alertView.title])
    {
        NSString *javascriptCmd = @"var divToChange = document.getElementById('labelDiv'); divToChange.innerText = 'Answered first alert'";
        [self.webView stringByEvaluatingJavaScriptFromString:javascriptCmd];
    }
    else
    if([@"Hello2" isEqualToString:alertView.title])
    {
        NSString *javascriptCmd = @"var divToChange = document.getElementById('labelDiv'); divToChange.innerText = 'Answered other alert'";
        [self.webView stringByEvaluatingJavaScriptFromString:javascriptCmd];
    }
}
```

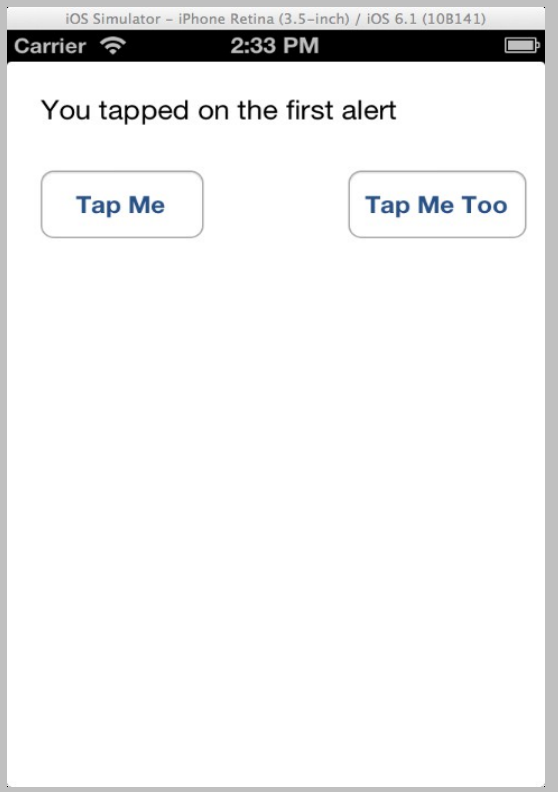
# Things to remember about hybrid apps:

- Any access to the device features (gyroscope, camera, media player, magnetometer, Bluetooth stack, etc) has to be done through the UIWebViewDelegate
- The more Javascript in the app, the slower it will go.
- The more changes the Javascript makes to the DOM, the s-l-o-w-e-r it will go, in general. Getting around this takes finagling and is not the same across platforms.
- UI widgets in HTML do not automatically look like the native UI widgets. They have to be styled to do it.
- *There is no such thing as a free lunch.*

# The same app in native code



# How native code talks to native UI

```
-(void)alertView:(UIAlertView *)alertView didDismissWithButtonIndex:(NSInteger)buttonIndex
{
    if([@"Hello" isEqualToString:alertView.title])
    {
        self.label.text = @"You tapped on the first alert";
    }
    else
    if([@"Hello2" isEqualToString:alertView.title])
    {
        self.label.text = @"You tapped on the other alert";
    }
}
-(IBAction)firstButtonTapped:(id)sender
{
    UIAlertView *helloAlert = [[UIAlertView alloc] initWithTitle:@"Hello" message:@"You pushed a button" delegate:self cancelButtonTitle:@"Yep" otherButtonTitles:nil];
    [helloAlert show];
}

-(IBAction)secondButtonTapped:(id)sender
{
    UIAlertView *helloAlert = [[UIAlertView alloc] initWithTitle:@"Hello2" message:@"You pushed the other button page" delegate:self cancelButtonTitle:@"Yep" otherButtonTitles:nil];
    [helloAlert show];
}
```

# Advantages native code has over hybrid code

- Controls and widgets look and feel native without a lot of futzing or a 3rd party import
- Faster, especially on iOS
- Direct access to hardware features without needing a 3rd party import
- Direct access to all system APIs and classes (NSString, for example)
- *Code is more secure.* Unless you are always fetching HTML/Javascript/CSS from a server like a web browser you are loading it from a cache directory on the device. That cache is visible with tools like iExplorer and javascript is in clear text, so if you store app ids or developer tokens there it's like leaving the front door key under the Welcome mat.