

Developing Mobile Websites

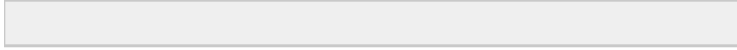
Consolidating the Web with Native Extensions

Lesson 1, Activity 2: Integrating with the Home Screen - Custom Icons

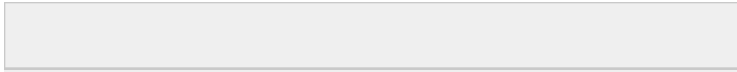
iPhone and other devices offer users the ability to add a shortcut/bookmark to a Web page on their home screen; our job as developers is to provide a custom icon that best represents our site or page. Including a line of code in the head of each page or including a specifically named icon in the Web site's root directory provides the functionality on iPhones and iPads; similar strategies work on Androids and other phones.

iPhone/iPad

1. Place a PNG icon, named `apple-touch-icon.png` or `apple-touch-icon-precomposed.png`, in the Web site root directory to specify the icon for the entire site if you don't want to add a link tag to pages; Safari on iOS won't add effects (rounded corners, etc.) to the precomposed icon.
2. Instead of adding an image to the Web site root directory, you can instead add a link element to the head section of any page to specify an icon for a single page:



3. To support different device resolutions (iPad vs. iPhone, for instance), add a size attribute to each link element:



The icon that is the most appropriate size for the device is used. If no sizes attribute is set, the element's size defaults to 57 x 57.

4. For iPhone and iPod touch, create icons that measure:

- 57 x 57 pixels
- 114 x 114 pixels (high resolution)

For iPad, create icons that measure:

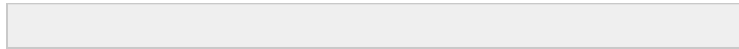
- 72 x 72 pixels
- 144 x 144 (high resolution)

5. The smallest icon larger than the recommended size is used if no icon matches the recommended size. If there are no icons larger than the recommended size, the largest icon is used. If multiple icons are suitable, the icon that has the precomposed keyword is used.
6. If no icons are specified using a link element, the root directory is searched for icons with the `apple-touch-icon` or `apple-touch-icon-precomposed` prefix.

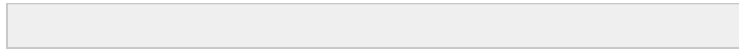
See the [Apple Developer](#) site for full details.

Other Devices


Android, Blackberry, and other devices have adopted the apple-touch-icon as a de facto standard. Please note that some versions of the Android require an absolute path, including the domain name, for the icon to work properly:



instead of



A Home Screen Icon for the *Jazz Calendar* Site

Open [ConsolidatingWebNativeExtensions/Demos/homescreen.html](#) from a mobile device. On an iPhone, tap the  icon, tap **Add to Home Screen**, and tap the upper-right **Add** button. The result should look like this:



Code Sample:

ConsolidatingWebNativeExtensions/Demos/homescreen.html

```
Jazz Calendar - Local Live Jazz Music Events
```

```
---- C O D E   O M I T T E D ----
```

We create a 57 x 57 pixel PNG image, save it in the images directory, and use

Hiding Browser UI

On mobile, screen real estate is at a premium; anything we can do to avoid wasting a few pixels of space on less-needed purposes can go a long way toward improving the user experience. A common strategy is to hide the address bar of the browser.

Some mobile browsers (WebOS, Blackberry) do this by default. Manual hiding for those browsers which do not already hide the address bar by default is possible only on the iPhone and Android - but, given the large market share of these devices, it is well worth doing it for those two.

We'll add some javascript to the head of our pages to scroll the page down and, thus, to hide the address bar on Android and iPhone.

The following JavaScript code, when placed inside a `<script>` tag in the head of our pages, effects the scroll-down on iPhones:

```
if( !window.location.hash && window.addEventListener ) {
  //when the page loads...
  window.addEventListener("load",function() {
    setTimeout(function() {
      //...scroll to top of page, hiding the address bar:
      window.scrollTo(0, 0);
    }, 0);
  });
}
```

The code checks to see if the browser supports the `addEventListener` function; if so, the code checks for the page to load and then scrolls to the top of the page, hiding the address bar

by passing 0 for the `x` coordinate and 0 for the `y` coordinate of the `scrollTo` function. The code also checks (`!window.location.hash`) to make sure the current URL doesn't include a hash; this would mean the user should be directed to a location below the top of the page - and thus we shouldn't scroll to the very top of the page by default.

The browser on an Android works differently: we pass a 1 (instead of 0) value for the `y` coordinate of the `scrollTo` function to scroll to the top of the page. We can exploit this quirk of the Android's implementation of the `scrollTo` function: when we use JavaScript to scroll the page to 1 and ask the device for its current `scroll` value, Android (unlike other devices) will return a value of 0. We can use this idiosyncrasy to write some cross-browser code to scroll the page:

```
function getScrollTop() {
    return scrollTop = window.pageYOffset || document.compatMode === "CSS1Compat" && document.documentElement.scrollTop || document.body.scrollTop || 0;
}

//scroll works differently on Android - scroll to (0,1):
window.scrollTo(0, 1);
var scrollTop, bodycheck = setInterval(function() {
    //keep polling the DOM until loaded:
    if(document.body) {
        clearInterval( bodycheck );
        scrollTop = getScrollTop();
        //if scrollTop is 1, set scrollTop to 0, else set to 1
        window.scrollTo(0, scrollTop === 1 ? 0 : 1);
    }
}, 15 );
```

We define a function `getScrollTop` to test the top of the current scroll, relying on several different tests to cover our bases among different device/browser's handling of this property. We scroll to `y=1` with the initial call to `window.scrollTo`. Given that we need the body DOM element to be defined before executing this, we test (`bodycheck`) in 15 millisecond intervals until we see it has loaded. We then scroll to the appropriate `y` position.

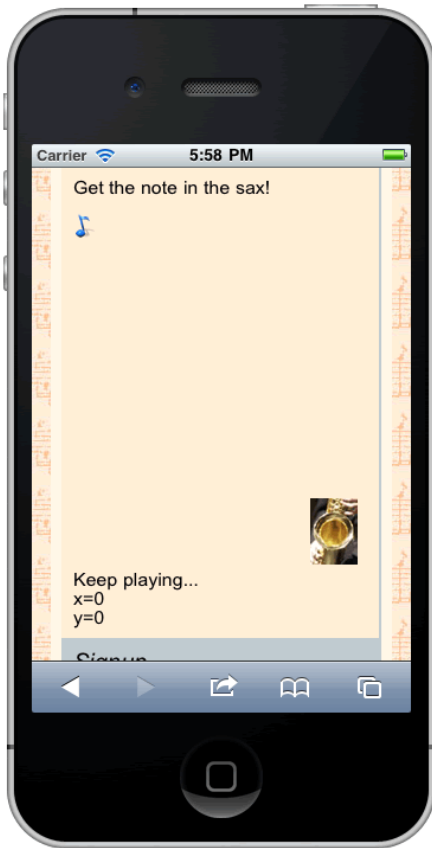
Open ConsolidatingWebNativeExtensions/Demos/hideaddress.html in a smartphone to check this in action.

See Scott Jehl's [excellent article on this topic](#) for more specifics.

Lesson 1, Activity 4: Using the Accelerometer

Both the iPhone and the Android expose accelerometer data from the phone through JavaScript. On the iPhone, the `window.ondevicemotion` event handler gives the device's current acceleration via the `accelerationIncludingGravity.x` and `accelerationIncludingGravity.y` values.

To illustrate the use of the accelerometer, let's write a simple game for the *Jazz Calendar* site: users are asked to tilt their iPhone to move the blue note into the sax at lower right:



Code Sample:

ConsolidatingWebNativeExtensions/Demos/accelerometer.html

```

---- CODE O M I T T E D ----

#no {
  display: none;
}
#note {
  position: absolute;
  top: 50px;
  left: 50px;
}
#saxbell {
  position: absolute;
  top: 240px;
  left: 200px;
}
#yes {
  position: relative;
  height: 300px;
  width: 245px;
}

---- CODE O M I T T E D ----

Get the note in the sax!

Your browser does not support Device Orientation and Motion API. Try this sample with iPhone, iPod or iPad with iOS 4.2+.

```

```
---- C O D E   O M I T T E D ----
```

We test whether `window.DeviceMotionEvent` is undefined; if not, we must be using a relatively recent iPhone and the needed JavaScript functionality is supported. `window.ondevicemotion = function(event)` gets the current state of the phone, with `event.accelerationIncludingGravity.x` and `event.accelerationIncludingGravity.y` exposing the x and y components of acceleration.

The `setInterval` function runs every 10 milliseconds, updating the CSS absolute position of the `#note` image. Physics 101 gives us velocity from acceleration and position from velocity; we update the CSS position of the `#note` image each time, checking to make sure it stays within the `#yes` div layer. We display the x and y position each time, along with a "You win!" message if the `#code` image lands on the sax. The Angry Birds folks probably aren't worried about our game competing with their franchise, but it illustrates the concept nicely. And thanks to the folks amenscher.com for the idea.

Lesson 1, Activity 6: Adding Home Screen Icon and Address-Bar Hiding to the *Pickup Soccer* Site

Duration: 10 to 20 minutes.

In this exercise, you will add a custom icon for a page on the Pickup Soccer site and add JavaScript to hide the address bar.

1. Open [ConsolidatingWebNativeExtensions/Exercises/soccer/index.html](#) in your file editor.
2. Add code in the head of the page to use [images/icon.png](#) as the home screen icon for the page.
3. Add JavaScript code between the existing script tags to hide the browser address bar.
4. Be sure to check your work on a smartphone or emulator.

Solution:

[ConsolidatingWebNativeExtensions/Solutions/soccer/index.html](#)

```
<!DOCTYPE html>
<html>
<head>
  <title>Soccer Pickup</title>
  <link rel="stylesheet" type="text/css" href="css/reset.css" />
  <link rel="stylesheet" type="text/css" href="css/style.css" />
  <meta name="viewport" content="initial-scale=1.0, width=device-width" />
  <link rel="apple-touch-icon" href="images/icon.png" />
  <script type="text/javascript">
    (function( win ){
      var doc = win.document;

      // If there's a hash, or addEventListener is undefined, stop here
      if( !location.hash && win.addEventListener ){

        //scroll to 1
        window.scrollTo(0, 1);
        var scrollTop = 1,
            getScrollTop = function(){
              return win.pageYOffset || doc.compatMode === "CSS1Compat" && doc.documentElement.scrollTop || doc.body.scrollTop || 0;
            }

        //reset to 0 on bodyready, if needed
        bodycheck = setInterval(function(){
          if(doc.body){
            clearInterval(bodycheck);
            scrollTop = getScrollTop();
            win.scrollTo(0, scrollTop === 1 ? 0 : 1);
          }
        }, 15 );

        win.addEventListener("load", function(){
          setTimeout(function(){
            //at load, if user hasn't scrolled more than 20 or so...
            if( getScrollTop() < 20 ){
              //reset to hide addr bar at onload
              win.scrollTo( 0, scrollTop === 1 ? 0 : 1 );
            }
          }, 0);
        } );
      }
    })(this);
  </script>
</head>

---- C O D E   O M I T T E D ----
```

We use `<link rel="apple-touch-icon" href="images/icon.png" />` to set the PNG file from the images directory as the icon associated with this page. We use the JavaScript code from the above example to hide the browser bar.

Lesson 1, Activity 7: PhoneGap

Accessing Camera, Contacts, Calendar, and More - But Not Yet

Lots of smartphone features, for instance the current location through the GeoLocation API and the accelerometer, are accessible via client-side JavaScript. But sadly, many other phone features - like the calendar, contacts, and camera - are not exposed directly via JavaScript.

One promising future development is the work of the World Wide Web Consortium's Device APIs Working Group. As their [Charter Web site](#) states, the mission of the group is to "create client-side APIs that enable the development of Web Applications that interact with device hardware, services and applications such as the camera, microphone, system sensors, native address books, calendars and native messaging applications." To date, these APIs are not widely available for production use.

PhoneGap

More phone features are accessible to native (rather than Web) apps - a native iPhone, Android, or Blackberry app can integrate with more features of their respective devices than can HTML/CSS/JavaScript Web apps. Of course, writing native apps requires a different skill set than does writing Web apps - a set of different skill sets, in fact, given that one must write for Android, for Blackberry, for iPhone, etc. What's a poor Web developer to do?

Go to [PhoneGap](#), an open-source mobile development framework. Either by downloading the PhoneGap SDK or, perhaps more intriguingly, uploading your code to PhoneGap for building for a variety of mobile platforms, you can leverage the same code across a variety of vendor-specific device APIs. In short, you are writing native phone apps with familiar (HTML, CSS, JavaScript) tools: PhoneGap exposes native mobile device APIs and data to JavaScript, offering write-once-run-anywhere development of "native" apps for all popular major platforms.

Getting Started

Our recommendation would be to start with [PhoneGap Build](#), which (in their words) allows you to "[s]imply write your app using HTML, CSS or JavaScript, upload it to the PhoneGap Build service and get back app-store ready apps for Apple iOS, Google Android, Palm, Symbian, BlackBerry and more."

1. Sign up for a [free developer account](#).
2. Confirm the signup (via the automatic email you'll receive) and login.
3. Once logged in, you can create your first app.

Your First App

1. Download the [PhoneGap sample app](#) from GitHub.
2. Login to PhoneGap Build.
3. Once logged in, from the [Your Apps page](#), choose **new app**.
4. Upload [index.html](#) from the sample app you downloaded.
5. PhoneGap Build will compile your app for the platforms listed.

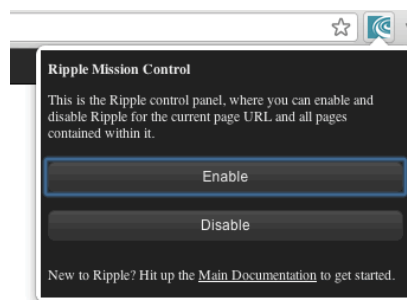
Of course it's not quite that easy: deploying to various device platforms requires certificate signing; a particularly onerous process for iOS devices. And we've not actually modified the sample app at all. But the power that PhoneGap (especially PhoneGap Build) offers - the ability to leverage familiar skills (JavaScript, HTML, CSS) and to access device-specific APIs for a wide range of disparate devices - is pretty impressive.

Ripple

A useful tool for developing and debugging PhoneGap applications is the [Ripple Mobile Environment Emulator](#) - an extension for the Google Chrome browser. While the emulator lacks some of the more specific properties of the various devices it simulates, it is a quick way to check out your PhoneGap apps.

If you have access to Google Chrome:

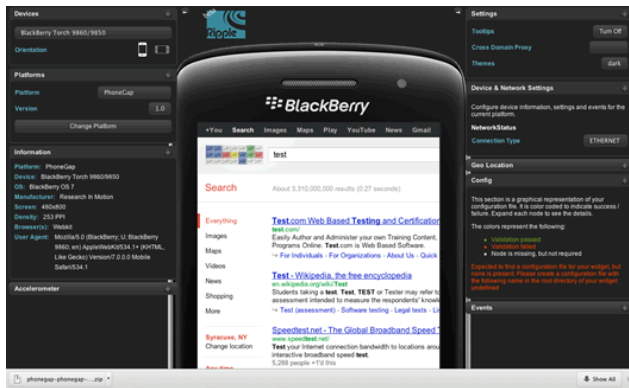
1. Install the [Ripple Extension](#).
- 2.



Enable Ripple by clicking on the Ripple Icon (right of the Chrome address bar)

3. Once Ripple is enabled, you can configure lots of options: choose the phone you wish to emulate, set your geolocation, and more.

Here's a screen shot of Ripple emulating a BlackBerry Torch 9860/9850:



You can use Ripple to test out your PhoneGap apps:

1. Choose "PhoneGap" for the Ripple "Platform" setting.
2. In Chrome, open [index.html](#) from the PhoneGap sample app you downloaded earlier - note that, if using a PC, you'll need to publish the PhoneGap app files to a Web server, rather than opening the file locally.
3. If not already, enable Ripple.

Here's the sample app with some slight modifications, with the relevant code from [index.html](#) below:



```
<div id="title_bar">Webucator Test App</div>

<div id="welcome" class="view">
  <div class="logo"></div>
  <div class="app_button" id="map_button">Show My Location</div>
  <div class="app_button" id="settings_button">Settings</div>
</div>
```

We change the display title of the app to "Webucator Test App" in the .title_bar div, and add the Webucator logo above the two existing buttons.