Creating mobile Web applications with HTML 5, Part 1: Combine HTML 5, geolocation APIs, and Web services to create mobile mashups

Find and track location coordinates to use in various Web services

Michael Galpin
Software Architect
eBay

29 June 2010
(First published 06 May 2010)

In the first part of this five part series, you will tap into one of the most popular new technologies available to mobile Web applications: geolocation. High-end smartphones all have GPS built-in to them, and now you will learn how it can be used by a Web application. In this article you will learn how to use the various aspects of the geolocation standard and how to use it with some popular Web services to create an interesting mobile mashup.

View more content in this series

25 May 2010: Added links to Part 2 of this series in About this series, Summary, and Resources sections.

02 Jun 2010: Added links to Part 3 of this series in About this series, Summary, and Resources sections.

08 Jun 2010: Added links to Part 4 of this series in About this series, Summary, and Resources sections.

29 Jun 2010: Added links to Part 5 of this series in About this series, Summary, and Resources sections.

About this series

Other articles in this series

- Part 2: Unlock local storage for mobile Web apps with HTML 5
- Part 3: Make mobile Web applications work offline with HTML 5
- Part 4: Use Web Workers to speed up your mobile Web applications
- Part 5: Develop new visual UI features in HTML 5

HTML 5 is a very hyped technology, but with good reason. It promises to be a technological tipping point for bringing desktop application capabilities to the browser. As promising as it is for traditional browsers, it has even more potential for mobile browsers. Even better, the most popular mobile browsers have already adopted and implemented many significant parts of the HTML 5 specification. In this five-part series, you will take a closer look at several of those new technologies that are part of HTML 5, that can have a huge impact on mobile Web application development. In each part of this series you will develop a working mobile Web application showcasing an HTML 5 feature that can be used on modern mobile Web browsers, like the ones found on the iPhone and Android-based devices.

Prerequisites

Frequently used acronyms

- API: Application Programming Interface
- CSS: Cascading stylesheet
- GPS: Global Positioning System
- HTML: Hypertext Markup Language
- JSONP: JSON with padding
- SDK: Software Developer Kit
- UI: User Interface
- W3C: World Wide Web Consortium

In this article, you will develop Web applications using the latest Web technologies. Most of the code here is just HTML, JavaScript, and CSS—the core technologies of any Web developer. The most important thing you will need are browsers to test things on. For this article, it is highly recommended that you have Mozilla Firefox 3.5+, as it is a desktop browser that supports geolocation. Of course you must test on mobile browsers too, and you will want the latest iPhone and Android SDKs for those. In this article iPhone SDK 3.1.3 and Android SDK 2.1 were used. See the Resources section for links.

The basics: Getting a fix

Geolocation by itself is somewhat of a novelty. It allows you to determine where the user is. However, just knowing this and reporting it to the user would not be very useful. Why would anyone care about their exact latitude and longitude? It is when you start using this in combination with other data and services that can make use of location, that you start to produce some interesting results. Almost all of these services will want a user's latitude and longitude as part of their input. Often this is all you need, so let's look at how you get this. Listing 1 shows the standard JavaScript API for this.

Listing 1. Finding a user: getCurrentPosition

```
navigator.geolocation.getCurrentPosition(successCallback,
errorCallback, options);
```

This is the essential API for geolocation. For a large class of applications, this is the only thing you will ever need. The geolocation object is part of the standard navigator object. It has a couple of methods, but the most commonly used is `getCurrentPosition`. Accessing a user's position is an expensive operation (you might be reaching out to a satellite in space!) and it requires the consent of the user, thus it is an asynchronous operation. Its parameters are callback functions: one for success, and one for failure.

The success function will get passed a single parameter of type `Position`. This object will have two properties: a timestamp property and a property called coords of type `Coordinates`. A `Coordinates` object has several properties:

- latitude
- longitude
- altitude
- accuracy
- altitudeAccuracy
- heading
- speed

Not all of these properties will be available on all devices, except for latitude, longitude, and accuracy. If the geolocation API is supported and the device can resolve its location, you can count on having latitude, longitude, and accuracy.

Develop skills on this topic

This content is part of a progressive knowledge path for advancing your skills. See HTML5 fundamentals

The error `callback` function will get passed a single parameter of type `PositionError`. An instance of `PositionError` will have two properties: code and message. The message is device specific, and is useful for debugging. The code should have one of three values:

- `PERMISSION_DENIED (1)`
- `POSITION_UNAVAILABLE (2)`
- `TIMEOUT (3)`

Your application should rely on the code to display a friendly error message to the user.

Note, the W3C specification also allows for a third parameter of options. These include things like a timeout for how long it takes to determine the user's location. However, this is not supported on devices like the iPhone yet, so it's use is not advised. Now that you have looked at the API in detail, take a look at a simple example to use it.

Integrating with Twitter

These days the hello world of mashups is to use Twitter in some way. For your first example, you will use Twitter's search API. It supports searching for tweets within a given radius of a location. Listing 2 shows the local Twitter search.

Listing 2. Local Twitter search

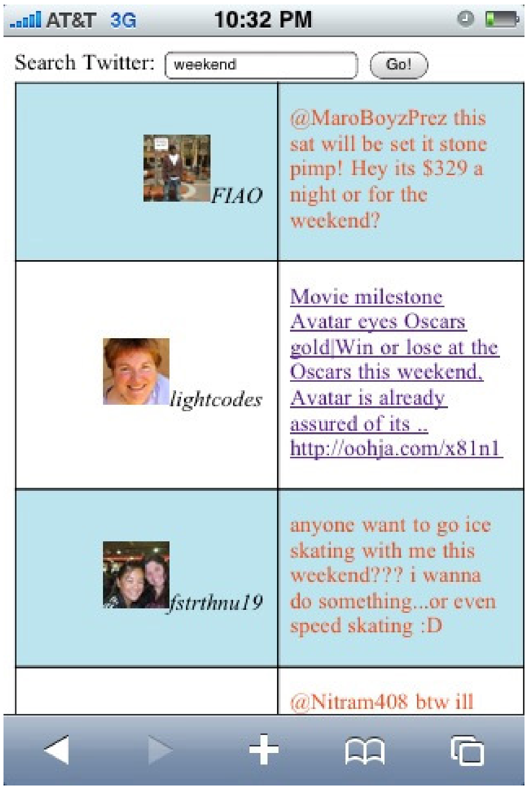
```
<!DOCTYPE html>
<html>
<head>
<meta name = "viewport" content = "width = device-width"/>
<title>Local Twitter Search</title>
<script type="text/javascript">
```

```
    function startSearch(){
        var gps = navigator.geolocation;
        if (gps){
            gps.getCurrentPosition(searchTwitter,
                   function(error){
                alert("Got an error, code: " + error.code + " message: "
+ error.message);
             });
        } else {
            searchTwitter();
        }
    }
    function searchTwitter(position){
        var query = "http://search.twitter.com/search.json?callback=showResults&q=";
        query += $("kwBox").value;
        if (position){
            var lat = position.coords.latitude;
            var long = position.coords.longitude;
            query += "&geocode=" + escape(lat + "," + long + ",50mi");
        }
        var script = document.createElement("script");
        script.src = query;
        document.getElementsByTagName("head")[0].appendChild(script);
    }
</script>
</head>
<body>
    <div id="main">
        <label for="kwBox">Search Twitter:</label>
        <input type="text" id="kwBox"/>
        <input type="button" value="Go!" onclick="startSearch()"/>
    </div>
    <div id="results">
    </div>
</body>
</html>
```

The user can enter a search term into the text box. A click on the button calls the `startSearch` function. This is where you use the geolocation API. First you check to see if it is available. If so, then you call the `getCurrentPosition` API. For the success callback, you use the function `searchTwitter`. For the error `callback` function, you pass a simple closure that simply displays the error information.

The `searchTwitter` function is called when the location is successfully determined by the browser. Here you use the position passed to the function to add a `geocode` parameter to your Twitter search query. The example in Listing 2 searches for tweets within 50 miles of the determined location. To call Twitter, you use a dynamic script tag, which is a technique often referred to as JSONP. This is supported by Twitter's search API, and allows you to call Twitter search directly from the browser, with no server needed at all. This is indicated by the `callback` parameter on the query. Notice it is set to `showResults`. That is the name of the function that will be invoked. It is not shown in Listing 2, since it is just used to create the UI, but it is available as part of the source code for this article (see Downloads). Figure 1 shows a screen capture of the code from Listing 2 running on an iPhone.

Figure 1. Search Twitter from an iPhone



FIAO	@MaroBoyzPrez this sat will be set it stone pimp! Hey its $329 a night or for the weekend?
lightcodes	Movie milestone Avatar eyes Oscars gold|Win or lose at the Oscars this weekend, Avatar is already assured of its .. http://oohja.com/x81n1
fstrthnu19	anyone want to go ice skating with me this weekend??? i wanna do something...or even speed skating :D
	@Nitram408 btw ill

This application, like many other location-aware applications, only needs to get the location one time. However, other apps will need to keep track of users as they are on the move. Those applications will need to use the other, more advanced geolocation APIs.

More advanced: Tracking

Sometimes your application needs not only the user's current location, but it needs to be updated each time the user changes location. There's an API for that and it's called `watchPosition`. It is very similar to `getCurrentPosition`, taking the same parameters. The one major difference is that it returns an ID. This ID can be used in conjunction with the final geolocation API, `clearWatch`. This function takes the ID you got from `watchPosition`. You see, when you call `watchPosition`, the browser will keep sending updates to the success callback function that you passed it, until you call `clearWatch`. Continuously getting a user's location can really drain the battery of a mobile device, so use these APIs with great caution. Now look at an example.

Integrating with Google Maps

For this example, you will leverage Google's Map APIs. These are optimized for use on mobile devices, with particular emphasis on the iPhone and Android platforms. This makes them very attractive to mobile Web developers, especially if you create location-aware applications. The sample application below will simply show the user's location on a map, updating the map each time the user's location changes. Listing 3 shows the mapping code.

Listing 3. Mapping application with Geolocation

```
<html>
```

```
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>I'm tracking you!</title>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?
     sensor=true"></script>
<script type="text/javascript">
    var trackerId = 0;
    var geocoder;
    var theUser = {};
    var map = {};
    function initialize() {
        geocoder = new google.maps.Geocoder();
        if (navigator.geolocation){
            var gps = navigator.geolocation;
            gps.getCurrentPosition(function(pos){
                var latLng = new google.maps.LatLng(pos.coords.
latitude,pos.coords.longitude);
                var opts = {zoom:12, center:latLng, mapTypeId:
google.maps.MapTypeId.ROADMAP};
                map = new google.maps.Map($("map_canvas"), opts);
                theUser = new google.maps.Marker({
                    position: latLng,
                    map: map,
                    title: "You!"
                });
                showLocation(pos);
            });
            trackerId = gps.watchPosition(function(pos){
                var latLng = new google.maps.LatLng(pos.coords.latitude,pos.
coords.longitude);
                map.setCenter(latLng);
                theUser.setPosition(latLng);
                showLocation(pos);
            });
        }
  }
</script>
</head>
<body style="margin:0px; padding:0px;" onload="initialize()">
    <div id="superbar">
        <span class="msg">Current location:
              <span id="location"></span>
          </span>
          <input type="button" value="Stop tracking me!"
onclick="stopTracking()"/>
      </div>
  <div id="map_canvas" style="width:100%; height:90%; float:left;
border: 1px solid black;">
  </div>
</body>
</html>
```

Once the body of the document loads, the `initialize` function is called. This function checks to see if geolocation is supported on the browser. If it is, then it calls `getCurrentPosition`, similar to the previous example in Listing 2. When it gets a location, it creates a map using the Google Map API. Notice how the latitude and longitude are used to create an instance of `google.maps.LatLng`. This object is used to center the map. Next, you create a marker on the map to represent the current location of the user. The marker once again uses the latitude and longitude that you received from the geolocation API.

After creating the map and putting a marker on it, you start tracking the user. You capture the ID returned from `watchPosition`. Whenever a new position is received, you re-center the map on the new location, and move the marker to that location. Listing 4 shows two more functions that you need to look at.

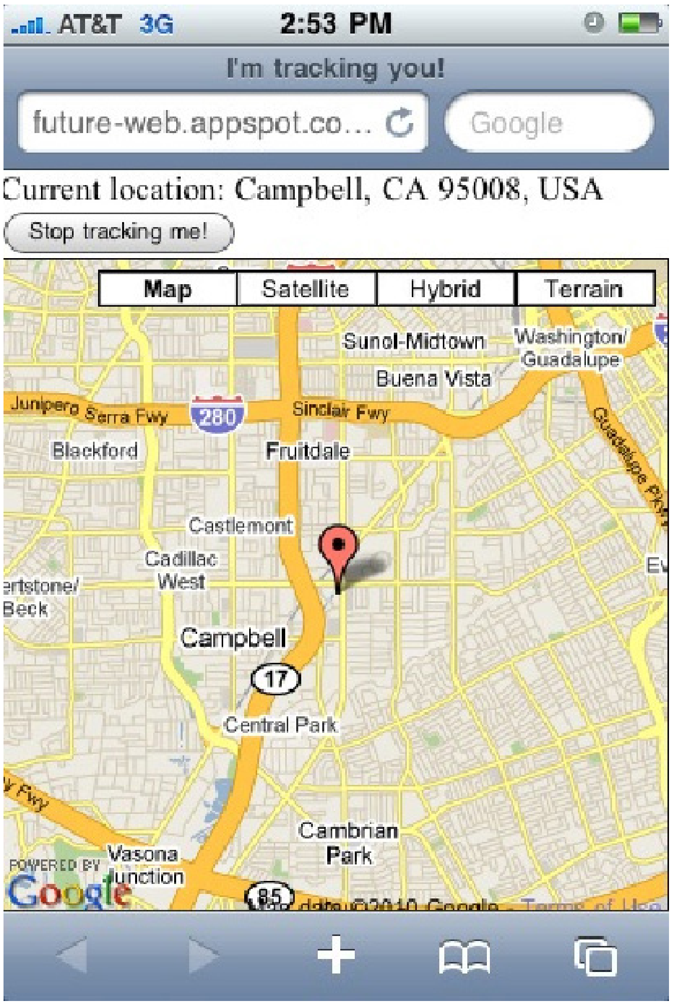
Listing 4. Geocoding and canceling tracking functions

```
function showLocation(pos){
    var latLng = new google.maps.LatLng(pos.coords.latitude,pos.coords.longitude);
    if (geocoder) {
        geocoder.geocode({'latLng': latLng}, function(results, status) {
          if (status == google.maps.GeocoderStatus.OK) {
            if (results[1]) {
                $("location").innerHTML = results[1].formatted_address;
            }
          }
        });
      }
}
function stopTracking(){
    if (trackerId){
        navigator.geolocation.clearWatch(trackerId);
    }
}
```

In Listing 3, the `showLocation` function is called when the map is initially drawn and when an update to the user's location is received. This function is shown in Listing 4. It uses an instance of `google.maps.Geocoder` (created at the beginning of the `initialize` function in Listing 3.) This API lets you perform geocoding, or, taking an address and turning it into mapping coordinates (latitude and longitude.) It also performs reverse-geocoding—taking mapping coordinates and returning a physical address. In this case, you take the coordinates produced by the geolocation API and use the Google Maps API to reverse geocode them. The results are then displayed on the screen.

The last function in Listing 4 is the `stopTracking` function. This is called when the user clicks on the button created in the HTML in Listing 3. Here you use the `trackerId` obtained when you first called the `watchPosition` function. You simply pass this to the `clearWatch` function and now the browser/device will stop getting the user's location and will also stop calling your JavaScript. Figure 2 shows a screen capture of the tracker application in use.

Figure 2. Tracking application



Of course, to really test out the tracking, you need to change the location. Using the Google App Engine can be a useful tool, since it allows you to easily upload your Web application to a publicly reachable location. Then you can test directly from your mobile device anywhere your device can get a good cellular connection. Once you do that, you can hop on public transportation or have somebody drive you around and watch your Web application respond to your changing location.

Summary

Other articles in this series

- Part 2: Unlock local storage for mobile Web applications with HTML 5
- Part 3: Make mobile Web applications work offline with HTML 5
- Part 4: Use Web Workers to speed up your mobile Web applications
- Part 5: Develop new visual UI features in HTML 5

This article has shown you how to use geolocation APIs in a mobile Web application. GPS can sound very sexy, but complicated. However, as you have seen, the W3C standard for geolocation provides a very simple API. It is straightforward to get a user's location and track that location over time. From there you can pass the coordinates to a variety of Web services that support location, or perhaps you have your own location aware service that you are developing. In Part 2 of this series on HTML 5 and mobile Web applications, you will look at how to take advantage of local storage to improve the performance of mobile Web applications.

Downloads

Description	Name	Size
Article source code	geo.zip	3KB

Resources

Learn

- Creating mobile Web applications with HTML 5, Part 2: Unlock local storage for mobile Web applications with HTML 5 (Michael Galpin, developerWorks, May 2010): Explore an important new feature in HTML 5 for wireless apps. With the standardization of local storage and a simple API, store large amounts of data on the client and improve performance.
- Creating mobile Web applications with HTML 5, Part 3: Make mobile Web applications work offline with HTML 5 (Michael Galpin, developerWorks, June 2010): Enable your application to function with or without an Internet connection and learn to detect when your application goes from offline to online and vice versa.
- Creating mobile Web applications with HTML 5, Part 4: Use Web Workers to speed up your mobile Web applications (Michael Galpin, developerWorks, June 2010): Web Workers bring multi-threading to Web applications. Learn to work with Web Workers and which tasks are most appropriate for them.
- Creating mobile Web applications with HTML 5, Part 5: Develop new visual UI features in HTML 5: Add Canvas, CSS3, and more semantic elements to mobile Web apps (Michael Galpin, developerWorks, June 2010): Provide full 2-D graphics in the browser with Canvas, the most eye-catching of the new UI capabilities in HTML 5. Learn to use Canvas and other visual elements in your mobile Web applications.
- Create Ajax applications for the mobile Web (Michael Galpin, developerWorks, March 2010): Explore how to use Ajax, a key part of any mobile Web application.
- New elements in HTML 5 (Elliotte Rusty Harold, developerWorks, August 2007): HTML 5 is not just about JavaScript. Read about some of the new markup in HTML 5.
- Android and iPhone browser wars, Part 1: WebKit to the rescue (Frank Ableson, developerWorks, December 2009): Do you like the mobile Web application approach using HTML 5 approach, but still want your application in the iPhone App Store and Android Market? See how you can get the best of both worlds in Part 1 of this two-part article series.
- Dive Into HTML 5: Check out this free book for a great look at HTML 5 detection techniques as well as the many features of HTML 5.
- Safari Reference Library: Keep this resource handy if you develop Web applications for the iPhone.
- W3C HTML 5 Specification (Working Draft, March 2010): Explore this definitive source on HTML 5.
- More articles by this author (Michael Galpin, developerWorks, April 2006-current): Read articles about XML, Eclipse, Apache Geronimo, Ajax, more Google APIs, and other technologies.
- My developerWorks: Personalize your developerWorks experience.
- IBM XML certification: Find out how you can become an IBM-Certified Developer in XML and related technologies.
- XML technical library: See the developerWorks XML Zone for a wide range of technical articles and tips, tutorials, standards, and IBM Redbooks.
- developerWorks technical events and webcasts: Stay current with technology in these sessions.

- developerWorks on Twitter: Join today to follow developerWorks tweets.
- developerWorks podcasts: Listen to interesting interviews and discussions for software developers.

Get products and technologies

- Modernizr project: Get a comprehensive utility for detecting HTML 5 features including like localStorage, Web Workers, applicationCache and others.
- The Android Developers Web site: Download the Android SDK, access the API reference, and get the latest news on Android.
- iPhone SDK: Get the latest iPhone SDK to develop iPad, iPhone and iPod touch applications.
- The Android Open Source Project: Get the open source code for the Android mobile platform.
- Download the Google App Engine SDK: Download Java™ and Python tools to build scalable Web applications using Google.
- IBM product evaluation versions: Download or explore the online trials in the IBM SOA Sandbox and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- XML zone discussion forums: Participate in any of several XML-related discussions.
- developerWorks blogs: Check out these blogs and get involved.

About the author

Michael Galpin

Michael Galpin is an architect at eBay and a frequent contributor to developerWorks. He has spoken at various technical conferences, including JavaOne, EclipseCon, and AjaxWorld. To get a preview of what his next project, follow @michaelg on Twitter.