Switch to mobile version

apache / **cordova-plugin-geolocation**

Mirror of Apache Cordova Plugin geolocation

296 commits | **17** branches | **49** releases | **47** contributors | Apache-2.0

Branch: **master** | New pull request | Find file | Clone or download

jcesarmobile committed Jan 10, 2018 Merge pull request #103 from jcesarmobile/CB-13748 … Latest commit e74c87a Jan 10, 2018

.github	CB-11917 - Remove pull request template checklist item: "iCLA has bee…	Oct 5, 2016
doc	CB-4596 Date objects are supposed to be DOMTimeStamp (s)	Oct 27, 2015
src	CB-13664: remove deprecated platforms	Dec 12, 2017
tests	Set VERSION to 4.0.2-dev (via coho)	Dec 28, 2017
www	CB-13664: remove deprecated platforms	Dec 12, 2017
.appveyor.yml	CB-12935: (windows) Enable paramedic builds on AppVeyor	Jun 30, 2017
.eslintrc.yml	CB-12895 : added eslint and removed jshint	Aug 3, 2017
.gitignore	CB-10636 Add JSHint for plugins	Feb 26, 2016
.travis.yml	CB-13748: Add build-tools-26.0.2 to travis	Jan 9, 2018
CONTRIBUTING.md	Actually fixing the contribute link.	Oct 23, 2015
LICENSE	[license] adding apache license file	Aug 6, 2013
NOTICE	Add NOTICE file	Feb 27, 2014
README.md	CB-13664: remove deprecated platforms	Dec 12, 2017
RELEASENOTES.md	Fix release notes (#102)	Dec 30, 2017
package.json	Set VERSION to 4.0.2-dev (via coho)	Dec 28, 2017
plugin.xml	Set VERSION to 4.0.2-dev (via coho)	Dec 28, 2017

README.md

title	description
Geolocation	Access GPS data.

AppVeyor	Travis CI
BUILD PASSING	build passing

cordova-plugin-geolocation

This plugin provides information about the device's location, such as latitude and longitude.

Common sources of location information include Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs. There is no guarantee that the API returns the device's actual location.

> To get a few ideas, check out the sample at the bottom of this page or go straight to the reference content.

This API is based on the W3C Geolocation API Specification, and only executes on devices that don't already provide an implementation.

WARNING: Collection and use of geolocation data raises important privacy issues. Your app's privacy policy should discuss how the app uses geolocation data, whether it is shared with any other parties, and the level of precision of the data (for example, coarse, fine, ZIP code level, etc.). Geolocation data is generally considered sensitive because it can reveal user's whereabouts and, if stored, the history of their travels. Therefore, in addition to the app's privacy policy, you should strongly consider providing a just-in-time notice before the app accesses geolocation data (if the device operating system doesn't do so already). That notice should provide the same information noted above, as well as obtaining the user's permission (e.g., by presenting choices for **OK** and **No Thanks**). For more information, please see the Privacy Guide.

This plugin defines a global `navigator.geolocation` object (for platforms where it is otherwise missing).

Although the object is in the global scope, features provided by this plugin are not available until after the `deviceready` event.

```
    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
        console.log("navigator.geolocation works well");
    }
```

Reference

Installation

This requires cordova 5.0+ (current stable 1.0.0)

```
cordova plugin add cordova-plugin-geolocation
```

Older versions of cordova can still install via the deprecated id (stale 0.3.12)

```
cordova plugin add org.apache.cordova.geolocation
```

It is also possible to install via repo url directly (unstable)

```
cordova plugin add https://github.com/apache/cordova-plugin-geolocation.git
```

Supported Platforms

- Android
- iOS
- Windows

Methods

- navigator.geolocation.getCurrentPosition
- navigator.geolocation.watchPosition
- navigator.geolocation.clearWatch

Objects (Read-Only)

- Position
- PositionError
- Coordinates

navigator.geolocation.getCurrentPosition

Returns the device's current position to the `geolocationSuccess` callback with a `Position` object as the parameter. If there is an error, the `geolocationError` callback is passed a `PositionError` object.

```
navigator.geolocation.getCurrentPosition(geolocationSuccess,
                                         [geolocationError],
                                         [geolocationOptions]);
```

Parameters

- **geolocationSuccess**: The callback that is passed the current position.
- **geolocationError**: *(Optional)* The callback that executes if an error occurs.
- **geolocationOptions**: *(Optional)* The geolocation options.

Example

```
    // onSuccess Callback
    // This method accepts a Position object, which contains the
    // current GPS coordinates
    //
    var onSuccess = function(position) {
        alert('Latitude: '          + position.coords.latitude          + '\n' +
              'Longitude: '         + position.coords.longitude         + '\n' +
              'Altitude: '          + position.coords.altitude          + '\n' +
              'Accuracy: '          + position.coords.accuracy          + '\n' +
              'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
              'Heading: '           + position.coords.heading           + '\n' +
              'Speed: '             + position.coords.speed             + '\n' +
              'Timestamp: '         + position.timestamp                + '\n');
    };

    // onError Callback receives a PositionError object
    //
    function onError(error) {
        alert('code: '    + error.code    + '\n' +
              'message: ' + error.message + '\n');
    }

    navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

iOS Quirks

Since iOS 10 it's mandatory to provide an usage description in the `info.plist` if trying to access privacy-sensitive data. When the system prompts the user to allow access, this usage description string will displayed as part of the permission dialog box, but if you didn't provide the usage description, the app will crash before showing the dialog. Also, Apple will reject apps that access private data but don't provide an usage description.

This plugins requires the following usage description:

- `NSLocationWhenInUseUsageDescription` describes the reason that the app accesses the user's location.

To add this entry into the `info.plist`, you can use the `edit-config` tag in the `config.xml` like this:

```
<edit-config target="NSLocationWhenInUseUsageDescription" file="*-Info.plist" mode="merge">
    <string>need location access to find things nearby</string>
</edit-config>
```

Android Quirks

If Geolocation service is turned off the `onError` callback is invoked after `timeout` interval (if specified). If `timeout` parameter is not specified then no callback is called.

navigator.geolocation.watchPosition

Returns the device's current position when a change in position is detected. When the device retrieves a new location, the `geolocationSuccess` callback executes with a `Position` object as the parameter. If there is an error, the `geolocationError` callback executes with a `PositionError` object as the parameter.

```
var watchId = navigator.geolocation.watchPosition(geolocationSuccess,
                                                  [geolocationError],
                                                  [geolocationOptions]);
```

Parameters

- **geolocationSuccess**: The callback that is passed the current position.
- **geolocationError**: (Optional) The callback that executes if an error occurs.
- **geolocationOptions**: (Optional) The geolocation options.

Returns

- **String**: returns a watch id that references the watch position interval. The watch id should be used with `navigator.geolocation.clearWatch` to stop watching for changes in position.

Example

```
    // onSuccess Callback
    //   This method accepts a `Position` object, which contains
    //   the current GPS coordinates
    //
    function onSuccess(position) {
        var element = document.getElementById('geolocation');
        element.innerHTML = 'Latitude: '  + position.coords.latitude      + '<br />' +
                            'Longitude: ' + position.coords.longitude     + '<br />' +
                            '<hr />'      + element.innerHTML;
    }

    // onError Callback receives a PositionError object
    //
    function onError(error) {
        alert('code: '    + error.code    + '\n' +
              'message: ' + error.message + '\n');
    }

    // Options: throw an error if no update is received every 30 seconds.
    //
    var watchID = navigator.geolocation.watchPosition(onSuccess, onError, { timeout: 30000 });
```

geolocationOptions

Optional parameters to customize the retrieval of the geolocation `Position`.

```
{ maximumAge: 3000, timeout: 5000, enableHighAccuracy: true };
```

Options

- **enableHighAccuracy**: Provides a hint that the application needs the best possible results. By default, the device attempts to retrieve a `Position` using network-based methods. Setting this property to `true` tells the framework to use more accurate methods, such as satellite positioning. *(Boolean)*

- **timeout**: The maximum length of time (milliseconds) that is allowed to pass from the call to `navigator.geolocation.getCurrentPosition` or `geolocation.watchPosition` until the corresponding `geolocationSuccess` callback executes. If the `geolocationSuccess` callback is not invoked within this time, the `geolocationError` callback is passed a `PositionError.TIMEOUT` error code. (Note that when used in conjunction with `geolocation.watchPosition`, the `geolocationError` callback could be called on an interval every `timeout` milliseconds!) *(Number)*

- **maximumAge**: Accept a cached position whose age is no greater than the specified time in milliseconds. *(Number)*

Android Quirks

If Geolocation service is turned off the `onError` callback is invoked after `timeout` interval (if specified). If `timeout` parameter is not specified then no callback is called.

navigator.geolocation.clearWatch

Stop watching for changes to the device's location referenced by the `watchID` parameter.

```
navigator.geolocation.clearWatch(watchID);
```

Parameters

- **watchID**: The id of the `watchPosition` interval to clear. (String)

Example

```
    // Options: watch for changes in position, and use the most
    // accurate position acquisition method available.
    //
    var watchID = navigator.geolocation.watchPosition(onSuccess, onError, { enableHighAccuracy: true })

    // ...later on...

    navigator.geolocation.clearWatch(watchID);
```

Position

Contains `Position` coordinates and timestamp, created by the geolocation API.

Properties

- **coords**: A set of geographic coordinates. *(Coordinates)*

- **timestamp**: Creation timestamp for `coords`. *(DOMTimeStamp)*

Coordinates

A `Coordinates` object is attached to a `Position` object that is available to callback functions in requests for the current position. It contains a set of properties that describe the geographic coordinates of a position.

Properties

- **latitude**: Latitude in decimal degrees. *(Number)*

- **longitude**: Longitude in decimal degrees. *(Number)*

- **altitude**: Height of the position in meters above the ellipsoid. *(Number)*

- **accuracy**: Accuracy level of the latitude and longitude coordinates in meters. *(Number)*

- **altitudeAccuracy**: Accuracy level of the altitude coordinate in meters. *(Number)*

- **heading**: Direction of travel, specified in degrees counting clockwise relative to the true north. *(Number)*

- **speed**: Current ground speed of the device, specified in meters per second. *(Number)*

Android Quirks

altitudeAccuracy: Not supported by Android devices, returning `null`.

PositionError

The `PositionError` object is passed to the `geolocationError` callback function when an error occurs with navigator.geolocation.

Properties

- **code**: One of the predefined error codes listed below.
- **message**: Error message describing the details of the error encountered.

Constants

- `PositionError.PERMISSION_DENIED`
  - Returned when users do not allow the app to retrieve position information. This is dependent on the platform.
- `PositionError.POSITION_UNAVAILABLE`
  - Returned when the device is unable to retrieve a position. In general, this means the device is not connected to a network or can't get a satellite fix.
- `PositionError.TIMEOUT`
  - Returned when the device is unable to retrieve a position within the time specified by the `timeout` included in `geolocationOptions`. When used with `navigator.geolocation.watchPosition`, this error could be repeatedly passed to the `geolocationError` callback every `timeout` milliseconds.

Sample: Get the weather, find stores, and see photos of things nearby with Geolocation

Use this plugin to help users find things near them such as Groupon deals, houses for sale, movies playing, sports and entertainment events and more.

Here's a "cookbook" of ideas to get you started. In the snippets below, we'll show you some basic ways to add these features to your app.

- Get your coordinates.
- Get the weather forecast.
- Receive updated weather forecasts as you drive around.
- See where you are on a map.
- Find stores near you.
- See pictures of things around you.

Get your geolocation coordinates

```
function getWeatherLocation() {

    navigator.geolocation.getCurrentPosition
    (onWeatherSuccess, onWeatherError, { enableHighAccuracy: true });
}
```

Get the weather forecast

```
// Success callback for get geo coordinates

var onWeatherSuccess = function (position) {

    Latitude = position.coords.latitude;
    Longitude = position.coords.longitude;
    getWeather(Latitude, Longitude);
}

// Get weather by using coordinates

function getWeather(latitude, longitude) {

    // Get a free key at http://openweathermap.org/. Replace the "Your_Key_Here" string with that key.
    var OpenWeatherAppKey = "Your_Key_Here";

    var queryString =
      'http://api.openweathermap.org/data/2.5/weather?lat='
      + latitude + '&lon=' + longitude + '&appid=' + OpenWeatherAppKey + '&units=imperial';

    $.getJSON(queryString, function (results) {

        if (results.weather.length) {

            $.getJSON(queryString, function (results) {

                if (results.weather.length) {

                    $('#description').text(results.name);
                    $('#temp').text(results.main.temp);
                    $('#wind').text(results.wind.speed);
                    $('#humidity').text(results.main.humidity);
                    $('#visibility').text(results.weather[0].main);

                    var sunriseDate = new Date(results.sys.sunrise);
                    $('#sunrise').text(sunriseDate.toLocaleTimeString());

                    var sunsetDate = new Date(results.sys.sunrise);
                    $('#sunset').text(sunsetDate.toLocaleTimeString());
                }

            });
        }
    }).fail(function () {
        console.log("error getting location");
    });
}

// Error callback

function onWeatherError(error) {
    console.log('code: ' + error.code + '\n' +
        'message: ' + error.message + '\n');
}
```

Receive updated weather forecasts as you drive around

```
// Watch your changing position

function watchWeatherPosition() {

    return navigator.geolocation.watchPosition
    (onWeatherWatchSuccess, onWeatherError, { enableHighAccuracy: true });
}

// Success callback for watching your changing position

var onWeatherWatchSuccess = function (position) {

    var updatedLatitude = position.coords.latitude;
    var updatedLongitude = position.coords.longitude;

    if (updatedLatitude != Latitude && updatedLongitude != Longitude) {

        Latitude = updatedLatitude;
        Longitude = updatedLongitude;

        // Calls function we defined earlier.
        getWeather(updatedLatitude, updatedLongitude);
    }
}
```

See where you are on a map

Both Bing and Google have map services. We'll use Google's. You'll need a key but it's free if you're just trying things out.

Add a reference to the **maps** service.

```
 <script src="https://maps.googleapis.com/maps/api/js?key=Your_API_Key"></script>
```

Then, add code to use it.

```
var Latitude = undefined;
var Longitude = undefined;

// Get geo coordinates

function getMapLocation() {

    navigator.geolocation.getCurrentPosition
    (onMapSuccess, onMapError, { enableHighAccuracy: true });
}

// Success callback for get geo coordinates

var onMapSuccess = function (position) {

    Latitude = position.coords.latitude;
    Longitude = position.coords.longitude;

    getMap(Latitude, Longitude);

}

// Get map by using coordinates

function getMap(latitude, longitude) {

    var mapOptions = {
        center: new google.maps.LatLng(0, 0),
        zoom: 1,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };

    map = new google.maps.Map
    (document.getElementById("map"), mapOptions);


    var latLong = new google.maps.LatLng(latitude, longitude);

    var marker = new google.maps.Marker({
        position: latLong
    });

    marker.setMap(map);
    map.setZoom(15);
    map.setCenter(marker.getPosition());
}

// Success callback for watching your changing position

var onMapWatchSuccess = function (position) {

    var updatedLatitude = position.coords.latitude;
    var updatedLongitude = position.coords.longitude;

    if (updatedLatitude != Latitude && updatedLongitude != Longitude) {

        Latitude = updatedLatitude;
        Longitude = updatedLongitude;

        getMap(updatedLatitude, updatedLongitude);
    }
}

// Error callback

function onMapError(error) {
    console.log('code: ' + error.code + '\n' +
        'message: ' + error.message + '\n');
}

// Watch your changing position

function watchMapPosition() {

    return navigator.geolocation.watchPosition
    (onMapWatchSuccess, onMapError, { enableHighAccuracy: true });
}
```

Find stores near you

You can use the same Google key for this.

Add a reference to the **places** service.

```
<script src=
"https://maps.googleapis.com/maps/api/js?key=Your_API_Key&libraries=places">
</script>
```

Then, add code to use it.

```
var Map;
var Infowindow;
var Latitude = undefined;
var Longitude = undefined;

// Get geo coordinates

function getPlacesLocation() {
    navigator.geolocation.getCurrentPosition
    (onPlacesSuccess, onPlacesError, { enableHighAccuracy: true });
}

// Success callback for get geo coordinates

var onPlacesSuccess = function (position) {

    Latitude = position.coords.latitude;
    Longitude = position.coords.longitude;

    getPlaces(Latitude, Longitude);

}

// Get places by using coordinates

function getPlaces(latitude, longitude) {

    var latLong = new google.maps.LatLng(latitude, longitude);

    var mapOptions = {

        center: new google.maps.LatLng(latitude, longitude),
        zoom: 15,
        mapTypeId: google.maps.MapTypeId.ROADMAP

    };

    Map = new google.maps.Map(document.getElementById("places"), mapOptions);

    Infowindow = new google.maps.InfoWindow();

    var service = new google.maps.places.PlacesService(Map);
    service.nearbySearch({
        location: latLong,
        radius: 500,
        type: ['store']
    }, foundStoresCallback);

}

// Success callback for watching your changing position

var onPlacesWatchSuccess = function (position) {

    var updatedLatitude = position.coords.latitude;
    var updatedLongitude = position.coords.longitude;

    if (updatedLatitude != Latitude && updatedLongitude != Longitude) {

        Latitude = updatedLatitude;
        Longitude = updatedLongitude;

        getPlaces(updatedLatitude, updatedLongitude);
    }
}

// Success callback for locating stores in the area

function foundStoresCallback(results, status) {

    if (status === google.maps.places.PlacesServiceStatus.OK) {

        for (var i = 0; i < results.length; i++) {

            createMarker(results[i]);

        }
    }
}

// Place a pin for each store on the map

function createMarker(place) {

    var placeLoc = place.geometry.location;

    var marker = new google.maps.Marker({
        map: Map,
        position: place.geometry.location
    });

    google.maps.event.addListener(marker, 'click', function () {

        Infowindow.setContent(place.name);
        Infowindow.open(Map, this);

    });
}

// Error callback

function onPlacesError(error) {
    console.log('code: ' + error.code + '\n' +
        'message: ' + error.message + '\n');
}

// Watch your changing position

function watchPlacesPosition() {

    return navigator.geolocation.watchPosition
    (onPlacesWatchSuccess, onPlacesError, { enableHighAccuracy: true });
}
```

See pictures of things around you

Digital photos can contain geo coordinates that identify where the picture was taken.

Use Flickr API's to find pictures that folks have taken near you. Like Google services, you'll need a key, but it's free if you just want to try things out.

```
var Latitude = undefined;
var Longitude = undefined;

// Get geo coordinates

function getPicturesLocation() {

    navigator.geolocation.getCurrentPosition
    (onPicturesSuccess, onPicturesError, { enableHighAccuracy: true });

}

// Success callback for get geo coordinates

var onPicturesSuccess = function (position) {

    Latitude = position.coords.latitude;
    Longitude = position.coords.longitude;

    getPictures(Latitude, Longitude);
}

// Get pictures by using coordinates

function getPictures(latitude, longitude) {

    $('#pictures').empty();

    var queryString =
    "https://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=Your_API_Key&lat="
    + latitude + "&lon=" + longitude + "&format=json&jsoncallback=?";

    $.getJSON(queryString, function (results) {
        $.each(results.photos.photo, function (index, item) {

            var photoURL = "http://farm" + item.farm + ".static.flickr.com/" +
                item.server + "/" + item.id + "_" + item.secret + "_m.jpg";

            $('#pictures').append($("<img />").attr("src", photoURL));

           });
        }
    );
}

// Success callback for watching your changing position

var onPicturesWatchSuccess = function (position) {

    var updatedLatitude = position.coords.latitude;
    var updatedLongitude = position.coords.longitude;

    if (updatedLatitude != Latitude && updatedLongitude != Longitude) {

        Latitude = updatedLatitude;
        Longitude = updatedLongitude;

        getPictures(updatedLatitude, updatedLongitude);
    }
}

// Error callback

function onPicturesError(error) {

    console.log('code: ' + error.code + '\n' +
        'message: ' + error.message + '\n');
}

// Watch your changing position

function watchPicturePosition() {
        return navigator.geolocation.watchPosition
        (onPicturesWatchSuccess, onPicturesError, { enableHighAccuracy: true });
    }
```