Open *Source* | Open *Possibilities*

Proposed Modifications to the W3C Geolocation API

April 2013

QuIC
QUALCOMM INNOVATION CENTER, INC.

Introduction

- The Worldwide Web Consortium (W3C) introduced a geolocation API for Javascript developers
  - Initial work and implementations appeared in 2008
  - Basic specification has now been adopted by all major browser vendors
    - Available in underlying web runtime engines (Gecko, Trident, Webkit)
- The Geolocation Working Group in the W3C is responsible for the specification
  - Group has not progressed on specifications for more than a year
    - Shelved Geolocation Level 2 spec work
- W3C (esp. SysApps WG) member companies should come to a determination as to whether the current API is sufficient “for building Web applications with comparable capabilities to native applications” (SysApps WG charter)

Proposed Change Areas

- Mobile app developers have already had access to much richer location API's when compared to current web counterpart dating back to pre-smartphone days
  - BREW Iposdet and J2ME JSR-179 as examples
- Smartphone API's have improved upon this capability
  - Android Location Manager and Snapdragon SDK enhancements (https://developer.qualcomm.com/mobile-development/mobile-technologies/snapdragon-sdk-android/features/location)
- What is proposed currently are incremental changes to the W3C API
  - Geofencing
  - Indoor Location
- Future changes can examine additional capability to bridge the gap
- Current document may be found at: http://gmandyam.github.io/enhanced-geolocation/

Open *Source* | Open *Possibilities*

Overview of Current API

QUALCOMM INNOVATION CENTER, INC.

Existing Geolocation API

- The Javascript API is allows for the following capabilities
  - One-shot location
  - Position watcher
    - Process that returns an event when the implementation has detected a change in user position
  - Ability to set location accuracy desired
    - Currently limited – developer can enable high accuracy mode or not
  - Features considered previously and rejected
    - Reverse geocoding capability
      - » Ability to look up address based on lat/lon
    - Proximity alert
      - » Event returned when device is detected to be crossing a boundary defined by a circle

API Detail

```
interface Geolocation {
  void getCurrentPosition(in PositionCallback successCallback,
                          in optional PositionErrorCallback errorCallback,
                          in optional PositionOptions options);

  long watchPosition(in PositionCallback successCallback,
                     in optional PositionErrorCallback errorCallback,
                     in optional PositionOptions options);

  void clearWatch(in long watchId);
 };

interface PositionCallback {
  void handleEvent(in Position position);
 };

interface PositionErrorCallback {
  void handleEvent(in PositionError error);
 };
```

API Detail (cont.)

- wacthPosition and getCurrentLocation take as an argument a PositionOptions object

```
interface PositionOptions {
        attribute boolean enableHighAccuracy;
        attribute long timeout;
        attribute long maximumAge;
};
```

- Both return a position object

```
interface Position {
   readonly attribute Coordinates? coords;
   readonly attribute Address? address;
   readonly attribute DOMTimeStamp timestamp;
 };
```

Open *Source* | Open *Possibilities*

Geofencing Modifications

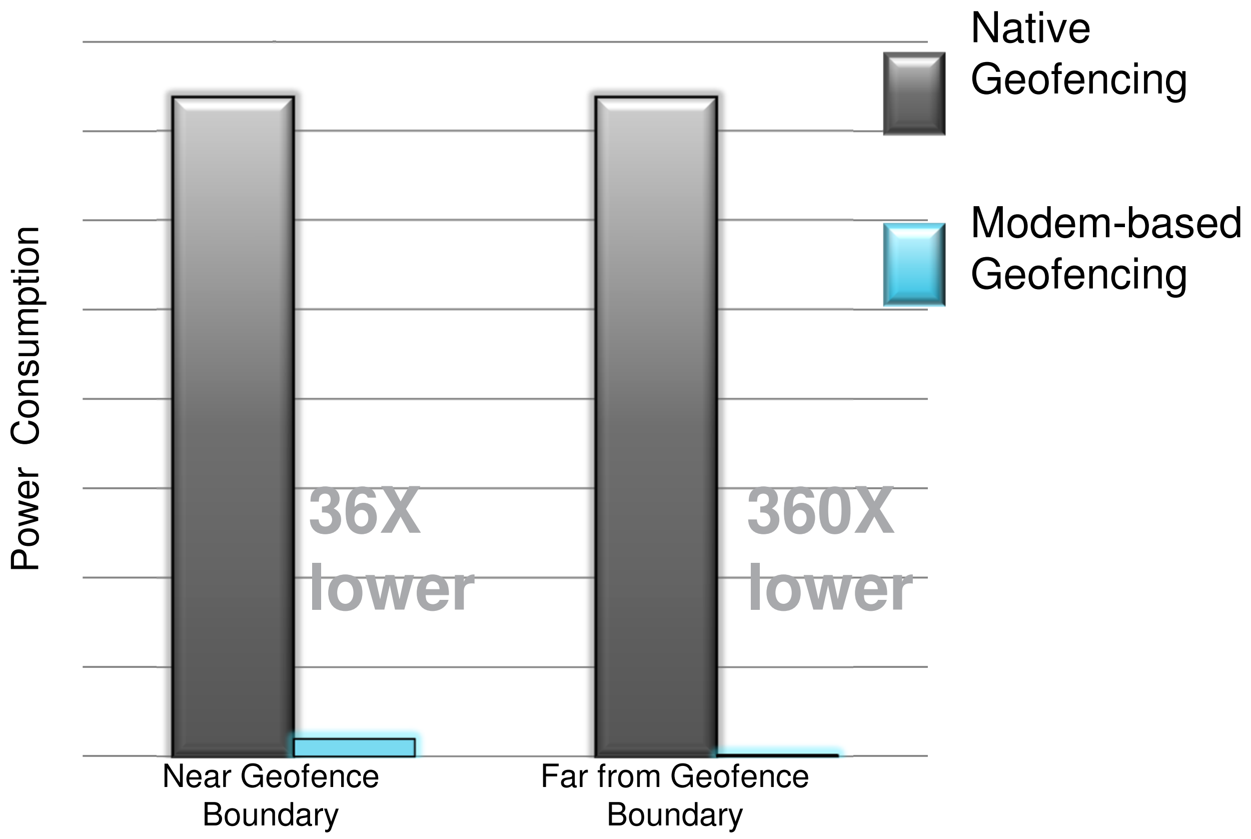
QUALCOMM INNOVATION CENTER, INC.

Introduction

- Current Geolocation API does not have any kind of geofencing ability
  - Typical geofencing capability would include defining a geofence with a centroid (i.e. lat, lon pair) and radius
- Justification is that it is simple to develop a geofencing method in Javascript leveraging the existing API
- For mobile devices, particularly multi-core implementations, this is not only limiting but can be detrimental to performance
  - CPU/GPU/Modem partitioning
  - Running geofencing processes on modem is significantly less power consuming then at the app level (e.g. JS)

Geofencing on Modem Versus Apps Processor

Optimizes responsiveness with much lower power



QuIC Proposed Changes for Geofencing Capability

- Introduces watchProximity and getCurrentProximityStatus, which provide for monitoring of breach of events based on a circular fence

```
void getCurrentProximityStatus(in CurrentProximityCallback successCallback,
                               in optional PositionErrorCallback errorCallback,
                               in watchProximitySettings settings,
                               in optional PositionOptions options);

long watchProximity(in ProximityCallback successCallback,
                               in optional PositionErrorCallback errorCallback,
                               in watchProximitySettings settings,
                               in optional PositionOptions options);
```

- watchProximitySettings defines the geofence

```
interface watchProximitySettings {
   attribute double latitude;
   attribute double longitude;
   attribute double radius; // in meters
   attribute long expiration; // in milliseconds
 };
```

QuIC Proposed Changes for Geofencing Capability (cont.)

- In addition to a position object, a ProximityState object is also returned by the functions that defines the device state with respect to the geofence

```
interface ProximityState {
  const unsigned short ENTERING_INSIDE = 0;
  const unsigned short LEAVING_OUTSIDE = 1;
  const unsigned short UNKNOWN = 3;
  readonly attribute unsigned short breachDirection;
};
```

  - breachDirection is interpreted differently depending on whether it is returned from getCurrentProximityStatus or watchProximity
    - Describes whether the device is inside or outside the geofence if it is returned as a result of a call to getCurrentProximityStatus
    - Describes the direction of motion (entering or leaving) if the device crosses the geofence boundary if it is returned as a result of a call to watchProximity

Code Sample

```
function updateDisplay(position, proxState) {
// Displays geofence event
if (proxState.breachDirection == 0) {
        document.write(“Entered geofence at “ + position.coords.latitude + “, “ + position.coords.longitude
        + “<br>”);}
else {
        if (proxState.breachDirection == 1) {
                document.write(“Exited geofence at “ + position.coords.latitude + “, “ + position.coords.longitude
                + “<br>”);}
        else {
                document.write (“Status unknown” + “<br>”);}}
        }

// Set geofence
var gf = {};
gf.latitude = 32.895732;
gf.longitude = -117.195861;
gf.radius = 10.5;
gf.timeout = 80000;
// Request geofence
var watchProximityId = navigator.geolocation.watchProximity(updateDisplay,gf);
function buttonClickHandler() {
// Cancel the geofence when the user clicks a button.
        navigator.geolocation.clearWatch(watchProximityId);    }
```

Indoor Location Enhancements

QuIC QUALCOMM INNOVATION CENTER, INC.

Indoor Location Enhancements to API

- Indoor location capability is now nearly ubiquitously-supported in smartphone hardware
- The underlying implementation should make the decision as to which location technology to use given current operating conditions
  - Setting enableHighAccuracy flag should result in indoor location mechanism being invoked if platform has indoor location capability and operating conditions allow for indoor location determination
- Web app should be able to leverage indoor location metadata when indoor location supported by platform and enabled
  - Floor number (first, second, third, etc.)
  - Additional building information (e.g. maps) that could assist in visualization

Proposal

- Enhance returned position object with optional readonly attributes

```
interface Position {
  readonly attribute Coordinates coords;
  readonly attribute Address address;
  readonly attribute DOMTimeStamp timestamp;
  readonly attribute double floorNumber;
  readonly any mapInfo;
  readonly attribute DOMString venueID;
};
```

- floorNumber is double type because of fractional floor representation
  - e.g. 1.5 is a mezzanine floor
- mapInfo defines a JSON object that could include building map data
  - Example: {"mapProvider": "Bing", "mapURL": "http://someplace.anywhere"}
- venueID specifies the ID. Can be used with MapInfo to download maps
  - Example: “White House”

Recommendation

- Ideally a common location API would be leveraged for both websites and packaged apps
- Current API however cannot bridge the gap between Web Apps and native apps
- Nevertheless, the privacy model is different and it could make sense for SysApps to take up extensions to the Geolocation API for installed apps
  - Could consider special privilege for enhanced geolocation capabilities
- Recommendation: SysApps WG take up Geolocation enhancements for packaged apps in Phase 2