



Clipcode's Guide to the W3C Geolocation API

Specification

<http://www.w3.org/TR/geolocation-API/>

Compatibility

<http://caniuse.com/#feat=geolocation>

Overview

The W3C Geolocation API is a simple API implemented by modern web browsers that supplies javascript code with the position of the user, either once-off or on a continuous basis.

The accuracy of the location information provided varies depending on the device's capabilities (some have GPS which provides accurate location, others try to generate location from the network edge in use, which is less accurate).

Concepts

The following concepts are inside the scope of the W3C Geolocation API:

- Coordinates – information such as longitude, latitude and (optionally) altitude
- Position – a combination of coordinates and a timestamp (when coordinates recorded)
- CurrentPosition – the position of the device now (once-off)
- Watch Position – keep a watch on the location of the device, and based on options, regularly update location information

The following concepts are outside the scope of the W3C Geolocation API, but are important for its real world usage:

- Map – Almost always you will want to use the position information to display it on a map. Map rendering is not part of the W3C Geolocation API, so you will have to use a specific mapping engine (Bing Maps, Google Maps) to deliver such functionality
- Storage of spatial information – The position information will often need to be stored in a database. This is also not part of the W3C Geolocation API. Such storage will be in the server backend, and will often involve a database that implements spatial features (such as SQL Server 2008 or “Denali”)

Object Model

The API (as defined in the W3C spec) has the following interfaces:

NavigatorGeolocation – implemented by the navigator, provides a property that returns the Geolocation object

```
[NoInterfaceObject]
interface NavigatorGeolocation {
    readonly attribute Geolocation geolocation;
};

Navigator implements NavigatorGeolocation;
```

Geolocation – takes three methods (getCurrentPosition – takes two callback parameters, one for success and one for error; watchPosition – takes two callback parameters, one for success and one for error, and takes an options parameter, PositionOptions; clearwatch – clears a previously set watch request)

```
[NoInterfaceObject]
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);
};

[Callback=FunctionOnly, NoInterfaceObject]
interface PositionCallback {
    void handleEvent(in Position position);
};

[Callback=FunctionOnly, NoInterfaceObject]
interface PositionErrorCallback {
    void handleEvent(in PositionError error);
};
```

PositionOptions has three attributes:

```
[Callback, NoInterfaceObject]
interface PositionOptions {
    attribute boolean enableHighAccuracy;
    attribute long timeout;
    attribute long maximumAge;
};
```

Position is a combination of coordinates and timestamp:

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

Coordinates is the actual location:

```
interface Coordinates {
    readonly attribute double latitude;
    readonly attribute double longitude;
    readonly attribute double? altitude;
    readonly attribute double accuracy;
    readonly attribute double? altitudeAccuracy;
    readonly attribute double? heading;
    readonly attribute double? speed;
};
```

PositionError is for error information:

```
interface PositionError {
  const unsigned short PERMISSION_DENIED = 1;
  const unsigned short POSITION_UNAVAILABLE = 2;
  const unsigned short TIMEOUT = 3;
  readonly attribute unsigned short code;
  readonly attribute DOMString message;
};
```

Security & Privacy

If you are a thief, and you know where a person lives, and you can entice that person into visiting your website (“special offer of X for people who live near Y”) and that person provides location information to you, then you know when that person is not at home and when it would be a good time for you to pay a visit.

Or imagine you work for the “security” services of a nasty dictatorial government and some democracy lovers are trying to organize – if you know which people are in certain locations at certain times, you know whom to arrest.

These are just two examples of how location information can be a very valuable commodity and users need to treat it with respect. Web browsers should (and most do) have specific user interface features to allow users to decide which sites should have access to their location information.

One security feature that some web browsers implement in relation to geolocation is to forbid files accessed locally (<file://>) from learning about geolocation information. To counter this, for your development machine only, there is usually a way to disable this feature (e.g. On Google Chrome – add this parameter to the command line: --allow-file-access-from-files).

Summary

As we have seen, the W3C Geolocation API is a simple to use API for accessing location information. Modern web applications should use it, as it is widely deployed, and when tied in with mapping visualization, it can greatly enhance a web application with little work. Due to privacy concerns, web application should be prepared to work when location information is not made available.