HTML - Geolocation API

HTML Geolocation API is used by web applications to access the geographical location of the user. Most modern browsers and mobile devices support the **Geolocation API**.

JavaScript can capture your latitude and longitude and can be sent to a backend web server and do fancy location-aware things like finding local businesses or showing your location on a map.

Syntax

```
var geolocation = navigator.geolocation;
```

The geolocation object is a service object that allows widgets to retrieve information about the geographic location of the device.

Geolocation API Methods

The Geolocation API provides the following methods:

Method	Description
getCurrentPosition()	This method retrieves the current geographic location of the user.
watchPosition()	This method retrieves periodic updates about the current geographic location of the device.
clearWatch()	This method cancels an ongoing watchPosition call.

Example

Following is a sample code to use any of the above methods:

```
function getLocation() {
   var geolocation = navigator.geolocation;
   geolocation.getCurrentPosition(showLocation, errorHandler);

   watchId = geolocation.watchPosition(showLocation, errorHandler, {
      enableHighAccuracy: true,
      timeout: 5000,
      maximumAge: 0
  });

  navigator.geolocation.clearWatch(watchId);
}
```

Here, **showLocation** and **errorHandler** are callback methods that would be used to get the actual position as explained in the next section and to handle errors if there are any.

Location Properties

Geolocation methods **getCurrentPosition()** and **getPositionUsingMethodName()** specify the callback method that retrieves the location information. These methods are called asynchronously with an object **Position** which stores the complete location information.

The **Position** object specifies the current geographic location of the device. The location is expressed as a set of geographic coordinates together with information about heading and speed.

The following table describes the properties of the Position object. For the optional properties, if the system cannot provide a value, the value of the property is set to null.

Property	Type	Description
coords	objects	Specifies the geographic location of the device. The location is expressed as a set of geographic coordinates together with information about heading and speed.
coords.latitude	Number	Specifies the latitude estimate in decimal degrees. The value range is [-90.00, +90.00].
coords.longitude	Number	Specifies the longitude estimate in decimal degrees. The value range is [-180.00, +180.00].
coords.altitude	Number	[Optional] Specifies the altitude estimate in meters above the WGS 84 ellipsoid.
coords.accuracy	Number	[Optional] Specifies the accuracy of the latitude and longitude estimates in meters.
coords.altitudeAccuracy	Number	[Optional] Specifies the accuracy of the altitude estimate in meters.
coords.heading	Number	[Optional] Specifies the device's current direction of movement in degrees counting clockwise relative to true north.
coords.speed	Number	[Optional] Specifies the device's current ground speed in meters per second.
timestamp	date	Specifies the time when the location information was retrieved and the Position object was created.

Example

Following is a sample code that makes use of the "**position**" object. Here, the **showLocation()** method is a callback method:

```
function showLocation( position ) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  ...
}
```

Handling Errors

Geolocation is complicated, and it is very much required to catch any error and handle it gracefully.

The geolocation methods **getCurrentPosition()** and **watchPosition()** make use of an error handler callback method that gives a PositionError object. This object has the following two properties:

Property	Type	Description
code	Number	Contains a numeric code for the error.
message	String	Contains a human-readable description of the error.

The following table describes the possible error codes returned in the PositionError object.

Code	Constant	Description
0	UNKNOWN_ERROR	The method failed to retrieve the location of the device due to an unknown error.
1	PERMISSION_DENIED	The method failed to retrieve the location of the device because the application does not have permission to use the Location Service.
2	POSITION_UNAVAILABLE	The location of the device could not be determined.
3	TIMEOUT	The method was unable to retrieve the location information within the specified maximum timeout interval.

Example

Following is a sample code that makes use of the **PositionError** object. Here **errorHandler** method is a callback method:

```
function errorHandler( err ) {
   if (err.code == 1) {
      // access is denied
```

```
    }
    ...
}
```

Position Options

Following is the actual syntax of the **getCurrentPosition()** method:

```
getCurrentPosition(callback, ErrorCallback, options)
```

Here, the third argument is the **PositionOptions** object, which specifies a set of options for retrieving the geographic location of the device.

Following are the options that can be specified as a third argument:

Property	Type	Description
enableHighAccuracy	Boolean	Specifies whether the widget wants to receive the most accurate location estimate possible. By default, this is false.
timeout	Number	The timeout property is the number of milliseconds your web application is willing to wait for a position.
maximumAge	Number	Specifies the expiry time in milliseconds for cached location information.

Example

Following is a sample code that shows how to use above-mentioned methods:

```
function getLocation() {
   var geolocation = navigator.geolocation;
   geolocation.getCurrentPosition(showLocation,
                                  errorHandler,
                                  {maximumAge: 75000});
}
```

Examples of HTML Geolocation API

Here are some examples that show how to access geolocation in HTML:

Get Current Location

The following code shows how to access the current location of your device using JavaScript and HTML.

</> Open Compiler

```
<!DOCTYPE html>
<html>
<head>
      <title>
         Geolocation API Example
      </title>
</head>
<body>

   <h2>Geolocation API Example</h2>
   <p id="demo">
      Click the button to get your coordinates:
   </p>
   <button onclick="getLocation()">
      Show Location
   </button>

   <script>
      var x = document.getElementById("demo");

      function getLocation() {
         if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition);
         } else {
            x.innerHTML =
            "Geolocation is not supported by this browser.";
         }
      }

      function showPosition(position) {
         x.innerHTML = "Latitude: " + position.coords.latitude +
         "<br>Longitude: " + position.coords.longitude;
      }

   </script>

</body>
</html>
```

Error Handling in Geolocation

Following is a sample code that shows how to use the above-mentioned methods:

Open Compiler

```
<!DOCTYPE html>
<html>
<head>
      <title>Geolocation API Example</title>
</head>

<body>
   <h2>Geolocation API Example</h2>
   <p id="demo">
      Turn off location service of your device,
      See how the error is handled.
   </p>
   <button onclick="getLocation()">
      Show Location
   </button>

   <script>
      var x = document.getElementById("demo");

      function getLocation() {
         if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition, showError);
         } else {
            x.innerHTML = "Geolocation is not supported by this browser.";
         }
      }

      function showPosition(position) {
         x.innerHTML = "Latitude: " + position.coords.latitude +
         "<br>Longitude: " + position.coords.longitude;
      }

      function showError(error) {
         switch(error.code) {
            case error.PERMISSION_DENIED:
                  x.innerHTML =
                  "User denied the request for Geolocation.";
                  break;
            case error.POSITION_UNAVAILABLE:
                  x.innerHTML =
                  "Location information is unavailable.";
                  break;
            case error.TIMEOUT:
                  x.innerHTML =
                  "The request to get user location timed out.";
                  break;
            case error.UNKNOWN_ERROR:
                  x.innerHTML =
                  "An unknown error occurred.";
                  break;
         }
      }
   </script>

</body>
</html>
```

Supported Browsers

API					
Geolocation	Yes	9.0	3.5	5.0	16.0