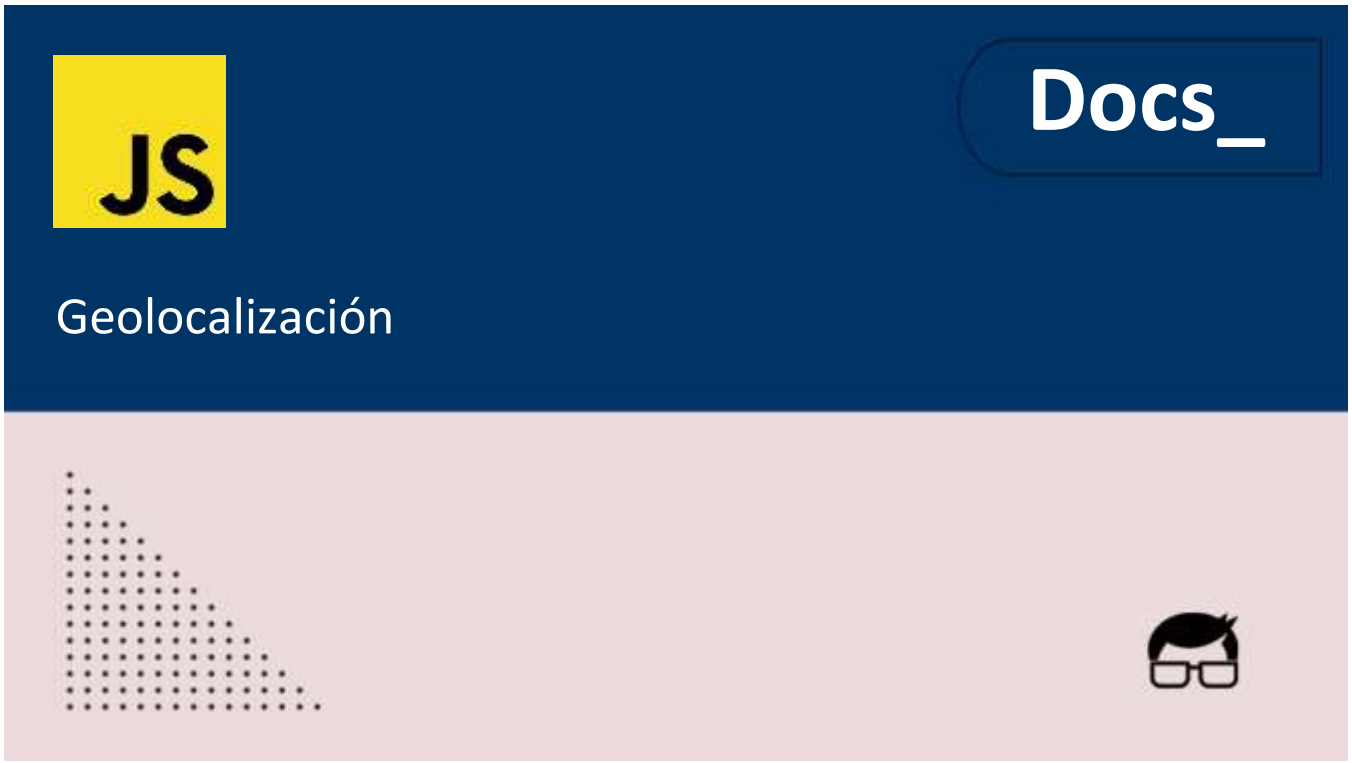

El API de geolocalización de HTML5 permite obtener la ubicación del usuario para usarlo en cualquier aplicación web. Como es normal, por motivos de privacidad el usuario debe de dar su consentimiento para poder usar esta información, por lo que si el usuario no quiere no podremos obtener esa información.

1. COMO COMPROBAR SI EL NAVEGADOR SOPORTA LA GEOLOCALIZACIÓN

Aunque los navegadores modernos lo soportan sin problemas nunca esta de más hacer esta comprobación para poder indicar al usuario si su navegador no soporta esta característica para que se de cuenta de que es problema de su navegador y no de la aplicación. Además es tan sencillo como hacer la siguiente comprobación:

```
if (navigator.geolocation) {
    // Obtenemos la ubicacion
} else {
    alert("Tu navegador no soporta la geolocalización, actualiza tu
navegador.");
}
```

2. COMO OBTENER LA UBICACIÓN DEL DISPOSITIVO

El primer método que veremos es getCurrentPosition() que como su nombre indica nos devuelve la posición actual. Esta función devuelve el valor de la posición tan pronto como le sea posible, por lo que puede devolver resultados con poca precisión, porque por ejemplo puede que se obtenga antes la posición por la IP, wifi o la red del móvil que por el GPS que puede tardar un rato hasta que encuentre una señal valida e incluso si tenemos una posición valida en la cache puede tomar esta con lo que la precisión puede ser bastante baja si estamos usando la aplicación en un móvil y nos estamos moviendo.

Por lo tanto esta función nos vale para obtener una ubicación rápida aunque no sea lo más preciso posible lo que es útil para aplicaciones en las que la ubicación exacta no sea muy importante.

La función getCurrentPosition() recibe como parámetro la función que se encargara de procesar la respuesta y opcionalmente puede recibir otra función para manejar posibles errores y/o un array de opciones. Vamos a ver como serán cada uno de estos parámetros, pero primero vamos a ver un ejemplo de código para entenderlo mejor.

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(mostrarPosicion, mostrarErrores,
opciones);
} else {
    alert("Tu navegador no soporta la geolocalización, actualiza tu navegador.");
}

function mostrarPosicion(posicion) {
    var latitud = posicion.coords.latitude;
    var longitud = posicion.coords.longitude;
    var precision = posicion.coords.accuracy;
    var fecha = new Date(posicion.timestamp);
    $('#posicion').append("
Latitud: " + latitud + "
");
    $('#posicion').append("
Longitud:" + longitud + "
");
    $('#posicion').append("
Precisión: " + precision + " metros
");
    $('#posicion').append("
Fecha: " + fecha + "
");
}

function mostrarErrores(error) {
    switch (error.code) {
        case error.PERMISSION_DENIED:
```

```
            alert('Permiso denegado por el usuario');
            break;
        case error.POSITION_UNAVAILABLE:
            alert('Posición no disponible');
            break;
        case error.TIMEOUT:
            alert('Tiempo de espera agotado');
            break;
        default:
            alert('Error de Geolocalización desconocido :' + error.code);
    }
}

var opciones = {
    enableHighAccuracy: true,
    timeout: 10000,
    maximumAge: 1000
};
```

Empecemos por la función mostrarPosicion que es la que se llama cuando se ha obtenido una posición valida. En esta función podemos acceder a la siguiente información:

posicion.coords.latitude: La latitud en formato decimal.

posicion.coords.longitude: La longitud en formato decimal.

posicion.coords.accuracy: La precisión de la posición medida en metros.

posicion.timestamp: La fecha y hora en la que se ha calculado la posición.

posicion.coords.altitude: La altitud sobre el nivel del mar medida en metros

posicion.coords.altitudeAccuracy: La precisión de la altitud.

posicion.coords.heading: La dirección en grados en el sentido de las agujas del reloj tomando el norte como 0 grados.

posicion.coords.speed: La velocidad en metros/segundo.

Los 4 primeros valores son devueltos siempre y el resto solo son devueltos en caso de que estén disponibles, por ejemplo si no estamos usando un GPS no tendremos altitud.

La función para controlar los errores es opcional pero en caso de usarla y de que ocurra algún error podemos controlar el error concreto o simplemente indicárselo al usuario. Los 3 posibles valores de errores son los del ejemplo anterior y el código es bastante claro por lo que no hace falta ninguna descripción adicional.

Finalmente también podemos indicar una lista de opciones, vamos a ver para que sirve cada una:

enableHighAccuracy: Activa o desactiva la opción de alta precisión.

timeout: Indica el tiempo máximo que se va a esperar para obtener una posición valida, antes de darlo por imposible.

maximumAge: Determina el tiempo máximo desde que se obtuvo la posicion en ms. Si lleva más tiempo en la cache no se usa.

3. COMO OBTENER LA UBICACIÓN CUANDO NOS MOVEMOS

Con la función anterior obtenemos la posición en la que estamos, pero si nos movemos esta posición seguirá estática a no ser que recarguemos la pagina. Para las situaciones en las que queremos que se calcule la posición continuamente para poder monitorizar el cambio tenemos la función watchPosition() que funciona igual que la función anterior pero mientras que con getCurrentPosition() solo se llama una única vez a la función que obtiene la posición con watchPosition() se llama continuamente a esa posición para poder reflejar los cambios en la posición ya sea porque el usuario se mueva o porque se pueda conseguir una señal más precisa.

La otra diferencia con getCurrentPosition() es que watchPosition() devuelve un identificador con el que podemos detener la función watchPosition() para que deje de actualizar la posición.

```
var watchId;

if (navigator.geolocation) {
    watchId = navigator.geolocation.watchPosition(mostrarPosicion,
mostrarErrores, opciones);
} else {
    alert("Tu navegador no soporta la geolocalización, actualiza tu
navegador.");
}

function mostrarPosicion(posicion) {
    var latitud = posicion.coords.latitude;
    var longitud = posicion.coords.longitude;
    var precision = posicion.coords.accuracy;
```

```
    var fecha = new Date(posicion.timestamp);
    $('#posicion').empty();
    $('#posicion').append("
Latitud: " + latitud + "
");
    $('#posicion').append("
Longitud:" + longitud + "
");
    $('#posicion').append("
Precisión: " + precision + " metros
");
    $('#posicion').append("
Fecha: " + fecha + "
");
}

function mostrarErrores(error) {
    switch (error.code) {
        case error.PERMISSION_DENIED:
            alert('Permiso denegado por el usuario');
            break;
        case error.POSITION_UNAVAILABLE:
            alert('Posición no disponible');
            break;
        case error.TIMEOUT:
            alert('Tiempo de espera agotado');
            break;
        default:
            alert('Error de Geolocalización desconocido :' + error.code);
    }
}

var opciones = {
    enableHighAccuracy: true,
    timeout: 10000,
    maximumAge: 1000
};

function detener() {
    navigator.geolocation.clearWatch(watchId);
}
```

Como se puede ver en este ejemplo el código es prácticamente igual ya que solo hay que cambiar el nombre de la función y en este caso obtenemos el identificador que devuelve para poder parar de actualizar la posición pulsando un botón por ejemplo.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Ejemplo Geolocalización</title>
    <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
    <script type="text/javascript">
    ...
    </script>
</head>
<body>
    <h1>Detección Geolocalización</h1>
    <div id="posicion"></div>
    <input type="button" value="Parar" onclick="detener();"/>
</body>
</html>
```

4. COMO MOSTRAR LA POSICIÓN EN GOOGLE MAPS

En los ejemplos anteriores hemos obtenido la posición que es lo que nos ofrece el API de geolocalización de HTML5 pero los datos por si solos no tiene mucho valor, o por lo menos para el usuario, por eso vamos a ver un ejemplo de como mostrar nuestra posición mediante la combinación de HTML5 y Google Maps.

Como la única diferencia es que ahora vamos a mostrar la posición en Google Maps, lo único que es necesario modificar es la función mostrarPosicion en la que tendremos que cargar el mapa.

```
var mapa = null;
var mapaMarcador = null;

function mostrarPosicion(posicion) {
    var latitud = posicion.coords.latitude;
    var longitud = posicion.coords.longitude;
    var precision = posicion.coords.accuracy;

    var miPosicion = new google.maps.LatLng(latitud, longitud);

    // Se comprueba si el mapa se ha cargado ya
    if (mapa == null) {
        // Crea el mapa y lo pone en el elemento del DOM con ID mapa
        var configuracion = {center: miPosicion, zoom: 16, mapTypeId:
google.maps.MapTypeId.HYBRID};
```

```
        mapa = new google.maps.Map(document.getElementById("mapa"),
configuracion);

        // Crea el marcador en la posicion actual
        mapaMarcador = new google.maps.Marker({position: miPosicion,
title:"Esta es tu posición"});
        mapaMarcador.setMap(mapa);
    } else {
        // Centra el mapa en la posicion actual
        mapa.panTo(miPosicion);
        // Pone el marcador para indicar la posicion
        mapaMarcador.setPosition(miPosicion);
    }
}
```

En la primera llamada a la función se crea el mapa y se coloca el marcador en la posición actual y en las siguientes llamadas se actualiza la posición del marcador y se centra el mapa para que la posición en la que nos encontramos siempre se muestre en el centro del mapa.

El código es sencillo, pero si colocamos esta función tal cual en el ejemplo anterior no se mostrará ningún mapa y es que nos falta añadir el javascript del API de Google Maps y también hay que tener cuidado de darle un ancho y un alto al div que va a contener el mapa porque si no tampoco se muestra.

Puedes ver el ejemplo en vivo directamente en este enlace o bien pulsar en edit on codepen, porque parece que por algún motivo no se puede ver directamente el ejemplo correctamente en la pestaña result.

Y con esto ya tenemos el ejemplo de geolocalización con HTML5 en Google Maps. Como siempre si quieres descargar el código de los 3 ejemplos puedes hacerlo desde aquí