GEOLOCATION

API Geolocation

Tres métodos específicos son provistos para usar la API:

getCurrentPosition(ubicación, error, configuración)

Este es el método utilizado para consultas individuales. Puede recibir hasta tres atributos:

- Una **función** para procesar la ubicación retornada.
- Una **función** para procesar los errores retornados.
- Un **objeto** para configurar cómo la información será adquirida.

Solo el primer atributo es **obligatorio.**

watchPosition(ubicación, error, configuración)

Este método es similar al anterior, excepto que comenzará un proceso de vigilancia para la detección de nuevas ubicaciones. Trabaja de forma similar que el conocido método **setInterval()** de Javascript, repitiendo el proceso automáticamente en determinados períodos de tiempo.

Retorna un valor que puede ser almacenado en una variable para luego ser usado como referencia por el método **clearWatch()** y así detener la vigilancia.

clearWatch(id)

Este método es similar a **clearInterval()** usado para detener los procesos comenzados por **setInterval().**

¿Cómo uso getCurrentPosition?

getCurrentPosition(ubicacion)

Este atributo es una **función** que recibirá un objeto llamado *Position*, el cual contiene toda la información retornada por los sistemas de ubicación.

El objeto **Position** tiene *dos* atributos:

- **coords:** Este atributo contiene un grupo de valores que establecen la ubicación del dispositivo y otros datos importantes.

- *latitude* (latitud)
- *longitude* (longitud)
- *altitude* (altitud en metros)
- *accuracy* (exactitud en metros)
- *altitudeAccuracy* (exactitud de la altitud en metros)
- *heading* (dirección en grados)
- *speed* (velocidad en metros por segundo).

- **timestamp**: Este atributo indica el momento en el que la información fue adquirida (en formato timestamp).

```
<section id="ubicacion">
    <button id="obtener">Obtener mi Ubicación</button>
</section>

function iniciar(){
    var boton=document.getElementById('obtener');
    boton.addEventListener('click', obtener, false);
}
function obtener(){
    navigator.geolocation.getCurrentPosition(mostrar);
}
```

```
function mostrar(posicion){
    var ubicacion=document.getElementById('ubicacion');
    var datos='';
    datos+='Latitud: '+posicion.coords.latitude+'<br>';
    datos+='Longitud: '+posicion.coords.longitude+'<br>';
    datos+='Exactitud: '+posicion.coords.accuracy+'mts.<br>';
    ubicacion.innerHTML=datos;
}
window.addEventListener('load', iniciar, false);
```

getCurrentPosition(ubicación, error)

Agregando un segundo atributo al método **getCurrentPosition(),** con otra **función**, podremos capturar los errores producidos en el proceso. Uno de esos errores ocurre cuando el usuario no acepta compartir sus datos.

El objeto **PositionError** tiene dos atributos internos, **error** y **message**, para proveer el valor y la descripción del error.

Los tres posibles errores son representados por las siguientes constantes:

- **PERMISSION_DENIED (permiso denegado)** Este error ocurre cuando el usuario no acepta activar el sistema de ubicación para compartir su información.
- **POSITION_UNAVAILABLE (ubicación no disponible)** Este error ocurre cuando la ubicación del dispositivo no pudo determinarse por ninguno de los sistemas de ubicación disponibles.
- **TIMEOUT (tiempo excedido)** Este error ocurre cuando la ubicación no pudo ser determinada en el período de tiempo máximo declarado en la configuración.

```
function obtener(){
    navigator.geolocation.getCurrentPosition(mostrar, errores);
}

function errores(error){
    alert('Error: ' + error.code + ' ' + error.message);
}
```

Podemos también controlar por errores de forma individual (error.PERMISSION_DENIED, por ejemplo) y actuar solo si ese error en particular ocurrió.

```
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;
    }
}
```

getCurrentPosition(ubicación, error, configuración)

El tercer atributo que podemos usar en el método **getCurrentPosition()** es un **objeto** conteniendo hasta tres posibles propiedades:

- **enableHighAccuracy** Esta es una propiedad booleana para informar al sistema que requerimos de la información más exacta que nos pueda ofrecer. Se consumen muchos recursos (batería), por lo que su uso debería estar limitado a circunstancias muy específicas. Para evitar consumos innecesarios, el valor por defecto de esta propiedad es false.
- **timeout** Esta propiedad indica el tiempo máximo de espera para que la operación finalice. Si la información de la ubicación no es obtenida antes del tiempo indicado, el error TIMEOUT es retornado. Su valor es en milisegundos.
- **maximumAge** Las ubicaciones encontradas previamente son almacenadas en un caché en el sistema. Si consideramos apropiado recurrir a la información grabada en lugar de intentar obtenerla desde el sistema (para evitar consumo de recursos o para una respuesta rápida), esta propiedad puede ser declarada con un tiempo límite específico. Si la última ubicación almacenada es más vieja que el valor de este atributo entonces una nueva ubicación es solicitada al sistema. Su valor es en milisegundos.

```
function obtener(){
    var geoconfig = {
        enableHighAccuracy: true,
        timeout: 10000,
        maximumAge: 60000
    };
navigator.geolocation.getCurrentPosition(mostrar, errores, geoconfig);
```

Este código intentará obtener la ubicación más exacta posible del dispositivo en no más de 10”, pero solo si no hay una ubicación previa en el caché capturada menos de 60“ atrás.

¿Cómo uso watchPosition?

watchPosition(ubicación, error, configuración)

Similar a **getCurrentPosition()**, el método **watchPosition()** recibe tres atributos y realiza la misma tarea: obtener la ubicación del dispositivo que está accediendo a la aplicación. La única diferencia es que el primero realiza una única operación, mientras que ofrece nuevos datos cada vez que la **watchPosition()** ubicación cambia. Este método vigilará todo el tiempo la ubicación y enviará información a la función correspondiente cuando se detecte una nueva ubicación, a menos que cancelemos el proceso con el método **clearWatch().**

Este es un ejemplo de cómo implementar el método **watchPosition()** basado en códigos previos:

```
control = navigator.geolocation.watchPosition(mostrar,errores,geoconfig);
```

No notará ningún cambio en un ordenador de escritorio usando este código, pero en un dispositivo móvil nueva información será mostrada cada vez que haya una modificación en la ubicación.

El valor retornado por el método **watchPosition()** fue almacenado en la variable control. Esta variable es como un identificador de la operación. Si más adelante queremos cancelar el proceso de vigilancia, solo debemos ejecutar la línea **clearWatch(control)** y **watchPosition()** dejará de actualizar la información.