


Géolocalisation en HTML5

Niveau : 



La géolocalisation fait partie des **API** gravitant autour de **HTML5** ([Geolocation API Specification](http://www.w3.org/TR/geolocation-API/) » <http://www.w3.org/TR/geolocation-API/>) et des nouvelles fonctionnalités introduites par la mobilité. Ses usages sont nombreux et souvent corrélés avec des bases de données de renseignements géographiques :

- Plans/cartes, calculs de position et d'itinéraires
- Renseignements locaux en mobilité (points d'intérêts proches)
- Résultats contextualisés sur les moteurs de recherche
- Méta-informations jointes aux photos/vidéos

Géolocalisation !

„Dis-moi où tu es, je te dirai ce que je peux te vendre”

De quelles informations a-t-on besoin pour se géolocaliser ?

Dans l'espace, trois coordonnées sont nécessaires :

- **Latitude** » <http://fr.wikipedia.org/wiki/Latitude>
- **Longitude** » <http://fr.wikipedia.org/wiki/Longitude>
- **Altitude** » <http://fr.wikipedia.org/wiki/Altitude> (facultative selon les besoins, la plupart des humains se déplaçant sur un même plan en deux dimensions à la surface de notre planète, exception faite des créatures souterraines)

Une quatrième coordonnée peut être prise en compte pour les puristes : le temps.

L'API donne aussi accès lorsqu'il y a déplacement à la vitesse et à la direction (en ° par rapport au nord géographique).

De quels moyens dispose un navigateur pour se géolocaliser ?



Différentes techniques sont mises à contribution avec plus ou moins de précision pour obtenir les coordonnées de géolocalisation. Elles peuvent être combinées pour affiner le résultat au cours du temps.

- Par satellite [GPS](http://fr.wikipedia.org/wiki/Global_Positioning_System) » http://fr.wikipedia.org/wiki/Global_Positioning_System (mobiles)
- Par triangulation GSM/3G (mobiles)
- Par triangulation WiFi (mobiles et bases de données adresses MAC)
- Par adresse IP (correspondance avec bases de données)

Si les deux premiers principes peuvent être mis en oeuvre directement par les plate-formes mobiles (smartphones et tablettes), les suivants dépendent de requêtes formulées à des bases de données et peuvent aussi être employées par les navigateurs classiques sur des postes fixes.



Par exemple, vous pourrez explorer du côté de Firefox l'URL appelée dans `about:config`, à la clé `geo.wifi.url`. Celle-ci correspond (actuellement) à un service de Google : <https://www.google.com/loc/json>

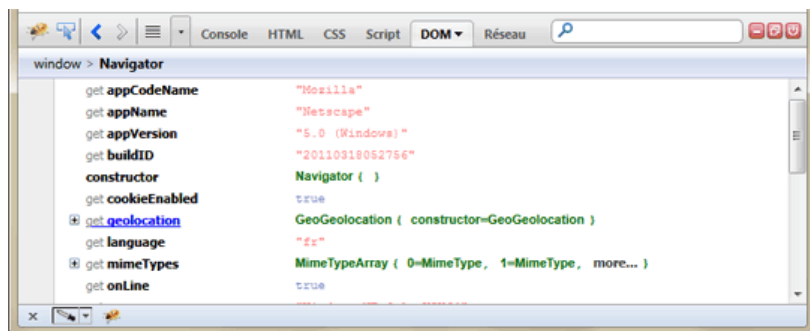
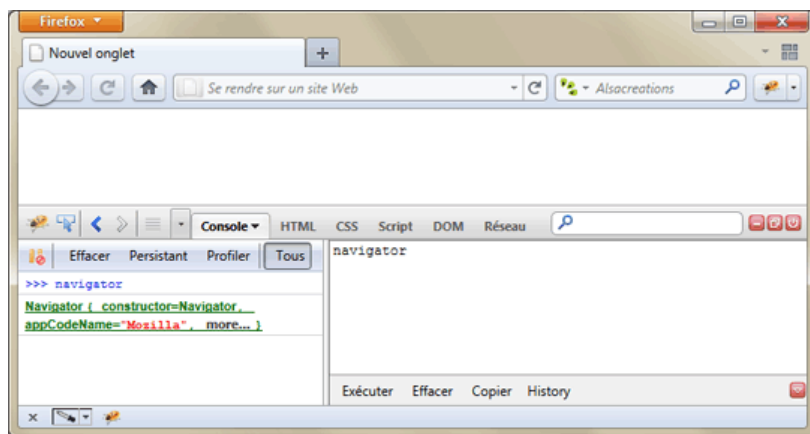
Certaines méthodes telles que le GPS sont plus lentes que d'autres et plus consommatrices en énergie notamment sur les mobiles. C'est pourquoi la première étape est souvent d'obtenir une approximation grossière - souvent avec une triangulation GSM - puis de mettre en route le module GPS au bout de quelques secondes (ceci se visualise sur iOS par la diminution progressive de l'étendue du cercle bleu de localisation).

Disponibilité de l'API ?

L'API repose sur l'objet `geolocation` membre de `navigator`.

```
if(navigator.geolocation) {  
    // L'API est disponible  
} else {  
    // Pas de support, proposer une alternative ?  
}
```

N'hésitez pas à explorer cet objet via la console de développement, disponible désormais sur tous les navigateurs modernes. Les captures d'écran ci-dessous correspondent à Firebug sous Firefox (F12), et sont similaires au résultat dans les extensions équivalentes pour Opera (Dragonfly), Chrome console JavaScript (Ctrl+Maj+J) ou même Internet Explorer.



Cette interface est équipée des méthodes `getCurrentPosition` et `watchPosition` qui seront détaillées par la suite. La détection de la disponibilité de l'API de géolocalisation peut aussi être menée avec [Modernizr](http://www.modernizr.com/) » <http://www.modernizr.com/> .

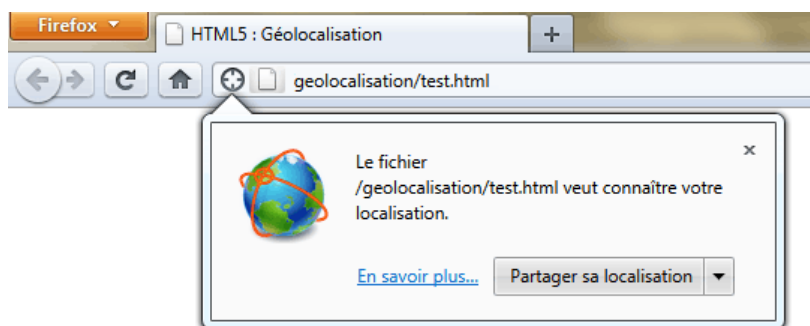
Appels asynchrones et fonctions de callback

La détection pouvant nécessiter un temps variable (et incertain), les appels aux méthodes de géolocalisation sont **asynchrones**. C'est-à-dire qu'ils retournent tout de suite pour donner la main à la suite du code, tout en déclenchant les techniques permettant de réunir toutes les informations nécessaires. Une fois ces informations obtenues de la part du navigateur, une fonction de *callback* définie par le développeur (vous) peut être appelée. Celle-ci recevra en argument toutes les valeurs réunies dans les propriétés d'un objet JavaScript.

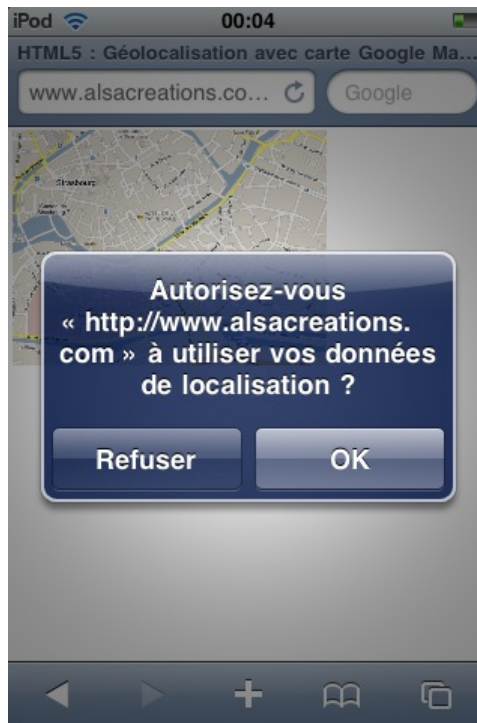
Retrouvez ici un article pour [Comprendre l'utilité des fonctions callback](http://www.epershand.net/developpement/algorithmie/explication-utilite-fonctions-callback) » <http://www.epershand.net/developpement/algorithmie/explication-utilite-fonctions-callback> .

Confidentialité

Pour des raisons évidentes de confidentialité, le navigateur ne communiquera pas vos coordonnées géographiques sans votre consentement explicite. Un appel aux deux fonctions précitées provoquera nécessairement un message d'avertissement pour l'utilisateur.



Les notifications peuvent revêtir des formes différentes, notamment entre navigateurs mobiles et *de bureau*.



Les doigts dans le code

Pour introduire la démonstration, débutons par une page HTML classique.

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8">
<title>HTML5 : Géolocalisation</title>
</head>
<body>

<!-- Un élément HTML pour recueillir l'affichage -->
<div id="infoposition"></div>

</body>
</html>
```

La géolocalisation repose sur deux méthodes :

- `getCurrentPosition` permettant un ciblage ponctuel
- `watchPosition` pour un suivi continu

`getCurrentPosition`

Nous récupérons grâce au code ci-dessous les coordonnées, la fonction "callback" est appelée directement à la suite du code et ainsi il ne reste plus qu'à récupérer et afficher le résultat grâce aux propriétés de l'objet JavaScript `position` :

`position.coords.latitude` pour la latitude, `position.coords.longitude` pour la longitude et enfin `position.coords.altitude` pour l'altitude.

Pour l'exemple, on passe en argument à `getCurrentPosition` une fonction nommée `maPosition` qui sera exécutée au moment opportun, lorsque les données seront disponibles.

```

<script>
function maPosition(position) {
    var infopos = "Position déterminée :\n";
    infopos += "Latitude : "+position.coords.latitude +"\n";
    infopos += "Longitude: "+position.coords.longitude+"\n";
    infopos += "Altitude : "+position.coords.altitude +"\n";
    document.getElementById("infoposition").innerHTML = infopos;
}

if(navigator.geolocation)
    navigator.geolocation.getCurrentPosition(maPosition);
</script>

```



Ici les exemples restent simples, nous allons voir plus loin que faire en cas d'erreur et comment coupler cette fonction avec l'API Google Maps.

[Exemple de carte avec getCurrentPosition\(\) » /xmedia/tuto/exemples/geolocalisation/map-getcurrentposition.html](#)

watchPosition

Pour suivre la position d'un élément, il suffit de remplacer `getCurrentPosition()` par la méthode `watchPosition()`.

```

// Fonction de callback en cas de succès
function surveillePosition(position) {
    var infopos = "Position déterminée :\n";
    infopos += "Latitude : "+position.coords.latitude +"\n";
    infopos += "Longitude: "+position.coords.longitude+"\n";
    infopos += "Altitude : "+position.coords.altitude +"\n";
    infopos += "Vitesse : "+position.coords.speed +"\n";
    document.getElementById("infoposition").innerHTML = infopos;
}

// On déclare la variable survId afin de pouvoir par la suite annuler le suivi de la
position
var survId = navigator.geolocation.watchPosition(surveillePosition);

```

Pour stopper ce suivi continu, il faut réexploiter la variable obtenue (qui est en quelque sorte un pointeur vers le processus de suivi) avec la méthode `clearWatch()`.

```
// Annule le suivi de la position si nécessaire.
navigator.geolocation.clearWatch(survId);
```

On aurait pu ne pas déclarer de fonction et faire comme pour l'exemple 1 de `getCurrentPosition()`. Mais il est toujours préférable d'avoir un code clair et détaillé.

 [Exemple de carte avec watchPosition\(\)](/xmedia/tuto/exemples/geolocalisation/map-watchposition.html) » </xmedia/tuto/exemples/geolocalisation/map-watchposition.html>

Gestion d'erreur

Les retours d'erreurs potentiels sont très importants et méritent d'être pris en compte dans toute application web. Ils permettent de savoir si l'utilisateur a refusé d'être géolocalisé, ou si la position n'a pu être obtenue.

```
// Fonction de callback en cas d'erreur
function erreurPosition(error) {
    var info = "Erreur lors de la géolocalisation : ";
    switch(error.code) {
        case error.TIMEOUT:
            info += "Timeout !";
            break;
        case error.PERMISSION_DENIED:
            info += "Vous n'avez pas donné la permission";
            break;
        case error.POSITION_UNAVAILABLE:
            info += "La position n'a pu être déterminée";
            break;
        case error.UNKNOWN_ERROR:
            info += "Erreur inconnue";
            break;
    }
    document.getElementById("infoposition").innerHTML = info;
```

Options

Diverses options peuvent être précisées pour configurer les appels :

- `enableHighAccuracy` : (true ou false > valeur par défaut) obtient une position plus précise via GPS
- `timeout` : (type long ou Infinity > valeur par défaut) durée avant renvoi vers la fonction d'erreur
- `maximumAge` : (type long ou Infinity, 0 > valeur par défaut) durée de la mise en cache de la position courante, si `maximumAge:0` alors la position ne viendra jamais du cache, elle sera toujours renouvelée

```
// Le paramètre maximumAge met en cache la position
// pour une durée de 600000 millisecondes (10 minutes),
// ainsi la position est mise à jour toutes les 10 minutes au maximum.
navigator.geolocation.getCurrentPosition(maPosition, erreurPosition,
{maximumAge:600000});
```

```
navigator.geolocation.getCurrentPosition(maPosition, erreurPosition,
{maximumAge:600000,enableHighAccuracy:true});
```

Exploiter les coordonnées

Une fois les informations obtenues, il y a potentiellement deux actions à entreprendre : exploiter directement les coordonnées sur une carte affichée sur la page, ou les mémoriser pour les valider avec un formulaire, ou les deux.

Soumettre les informations au serveur

Pour cela deux méthodes sont possibles :

Effectuer un appel AJAX avec jQuery :

```
$.post("http://www.votredomaine.com/position.php",
{lat:position.coords.latitude,lng:position.coords.longitude});
```

Cette méthode envoie en POST les résultats obtenus dans l'objet `position`.

Se servir de champs cachés :

On place, dans le code HTML, des champs cachés pour la latitude et la longitude

```
<input type="hidden" name="lat" id="lat" />
<input type="hidden" name="lng" id="lng" />
```

Dans le script jQuery, on attribue la valeur de la latitude et de la longitude dans les champs correspondants.

```
$("#lat").val(position.coords.latitude);
$("#lng").val(position.coords.longitude);
```

Puis l'on effectue la requête en PHP pour envoyer les données sur le serveur. Les données se retrouvent en entrée dans les tableaux `$_POST` ou `$_GET` selon la méthode d'envoi, ou plus simplement `$_REQUEST`.

```
echo $_REQUEST['lat'];
echo $_REQUEST['lng'];
```



Google Maps API

L'API Google Maps V3 » <http://code.google.com/intl/fr-FR/apis/maps/documentation/javascript/> est très aisée à exploiter en combinaison à la géolocalisation. Elle comprend de nombreuses fonctionnalités pour afficher une carte géographique, positionnée et équipée de marqueurs.

Vous aurez au préalable chargé l'API par une balise de script.

```
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?
sensor=set_to_true_or_false"></script>
```

Et préparé un élément HTML destiné à recevoir la carte.

```
<div id="map"></div>
```

Nous allons tout d'abord définir une position par défaut, puis quelques options pour l'affichage de la carte.

```
// Position par défaut (Châtelet à Paris)
```



```

var centerpos = new google.maps.LatLng(48.579400,7.7519);

// Options relatives à la carte
var optionsGmaps = {
    center:centerpos,
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    zoom: 15
};

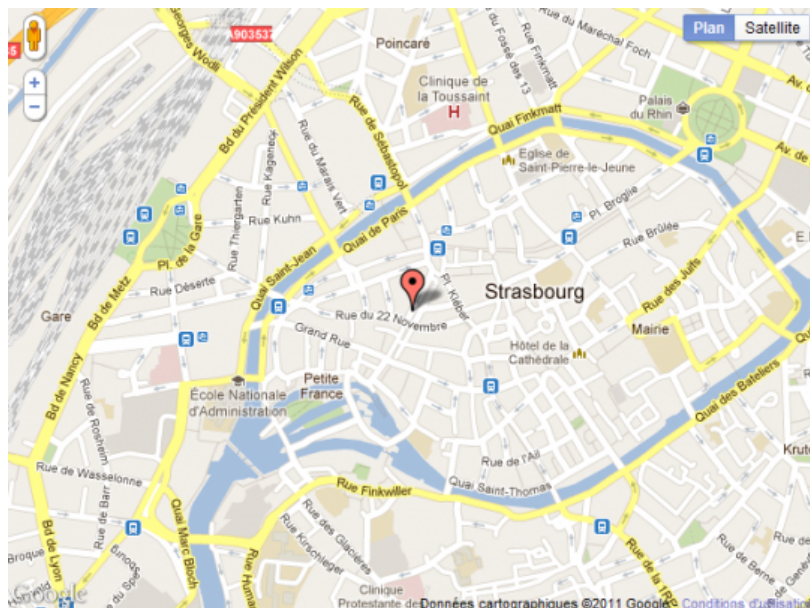
// ROADMAP peut être remplacé par SATELLITE, HYBRID ou TERRAIN
// Zoom : 0 = terre entière, 19 = au niveau de la rue

// Initialisation de la carte pour l'élément portant l'id "map"
var map = new google.maps.Map(document.getElementById("map"), optionsGmaps);

// .. et la variable qui va stocker les coordonnées
var latlng;

```

Aperçu de l'affichage :



Nous pouvons désormais passer à la suite, c'est ici qu'interviennent les fonctions callback et la gestion des erreurs, vues précédemment. La fonction `surveillePosition()`, correspondant au suivi de la position, est déclarée afin de récupérer les coordonnées.

```

// Fonction de callback en cas de succès
function maPosition(position) {

    var infopos = "Position déterminée :\n";
    infopos += "Latitude : "+position.coords.latitude +"\n";
    infopos += "Longitude : "+position.coords.longitude+"\n";
    infopos += "Altitude : "+position.coords.altitude +"\n";
    infopos += "Vitesse : "+position.coords.speed +"\n";
    document.getElementById("infoposition").innerHTML = infopos;

    // Un nouvel objet LatLng pour Google Maps avec les paramètres de position
    latlng = new google.maps.LatLng(position.coords.latitude,
    position.coords.longitude);

    // Ajout d'un marqueur à la position trouvée

```



```

var marker = new google.maps.Marker({
    position: latlng,
    map: map,
    title:"Vous êtes ici"
});

// Permet de centrer la carte sur la position latlng
map.panTo(latlng);

}

```

⚠ Sans oublier la gestion des erreurs !

```

// Fonction de callback en cas d'erreur
function erreurPosition(error) {
    var info = "Erreur lors de la géolocalisation : ";
    switch(error.code) {
        case error.TIMEOUT:
            info += "Timeout !";
            break;
        case error.PERMISSION_DENIED:
            info += "Vous n'avez pas donné la permission";
            break;
        case error.POSITION_UNAVAILABLE:
            info += "La position n'a pu être déterminée";
            break;
        case error.UNKNOWN_ERROR:
            info += "Erreur inconnue";
            break;
    }
    document.getElementById("infoposition").innerHTML = info;
}

```

Il ne vous reste plus qu'à "lancer" le suivi de la position, sans oublier la possibilité d'annuler le suivi.

```

if(navigator.geolocation) {
    survId = navigator.geolocation.getCurrentPosition(maPosition,erreurPosition);
} else {
    alert("Ce navigateur ne supporte pas la géolocalisation");
}

```

💡 **Bonus** : Personnaliser le marqueur est possible avec le paramètre `icon` passé à l'objet du constructeur `Marker()`.

```

marker = new google.maps.Marker({
    position: latlng,
    map: map,
    title:"Vous êtes ici",
    icon: "fleche.png"
});

```

