Géolocalisation HTML5 Javascript

- Qu'est-ce-que la géolocalisation ? A quoi ça sert ?
- Quelles sont les informations utiles ? Comment les obtenir ?
- Longitude, Latitude, Altitude
- Conversion d'unités
- Comment savoir si le navigateur supporte la géolocalisation ?
- Consentement de l'utilisateur
- Types de positionnement
  - Positionnement ponctuel
  - Suivi continu
- Autres informations détaillées

Qu'est-ce-que la géolocalisation ?

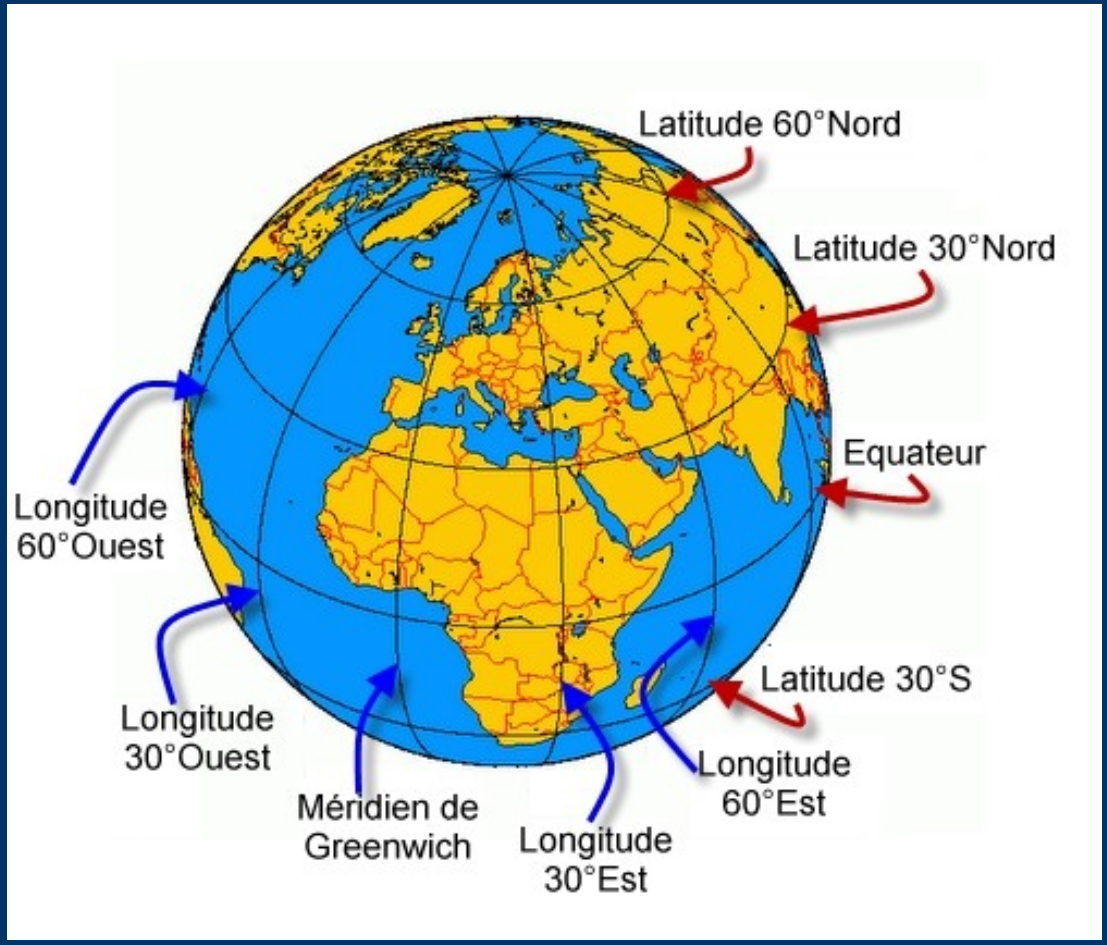
- La géolocalisation a été introduite dans **HTML5** pour réaliser des applications basées sur les **coordonnées géographiques**.
- C'est un procédé permettant de **positionner** un objet (une personne, etc.) sur un plan ou une carte à l'aide de ses coordonnées géographiques.
- Cette opération est réalisée à l'aide d'un **terminal** capable d'**être localisé** (GPS, GSM, WiFi,…)
- Voir aussi sur Wikipédia : géolocalisation

A quoi sert la géolocalisation ?

- Les applications de la géolocalisation sont en plein développement.
- **Objectifs** : gain de productivité, économies de carburant et de communications, sécurité accrue
- **Applications** : localisation de véhicules et de personnes, suivi en temps réel et historique, trajets détaillés sur carte, calcul des temps de conduite et de pause, surveillance de présence en entrée et/ou sorties de zones, localisation du véhicule le plus proche, d'un magasin, d'une station-service, d'un restaurant, génération de statistiques, marketing/vente/publicité ciblées, suivi de personnes (patients à risque, prisonniers, consommateurs,...)

Quelles informations sont utiles ?

- Sur la Terre, pour définir une position, on a besoin de **3 coordonnées** : la **latitude**, la **longitude** et l'**altitude**.



Latitude, longitude, altitude

- **La latitude** est le décalage en degrés par rapport à ***l'Equateur***. La latitude Nord a une valeur positive, la latitude Sud a une valeur négative.
- Exemples: Bruxelles (50.846583°), Moscou (55,754529°), New York (40.758637°), Rio de Janeiro (-22,968648°), Johannesbourg (-26,204299°)
- **La longitude** est le décalage en degrés par rapport au ***Méridien de Greenwich***. La longitude Est a une valeur positive, la longitude Ouest a une valeur négative.
- Exemples: Bruxelles (4,352694°), Moscou (37,619437°), New York (-73,985388°), Rio de Janeiro (-43,188282°), Johannesbourg (28,032033°)

Equateur et Méridien de Greenwich

Le Caire : 30°02′40″ Nord, 31°14′44″ Est => +30.04444°N, +31.24556°E

Sao Paulo : 23°32′52″ Sud, 46°38′11″ Ouest => -23.54778°S, -46.63639°O

Conversion d'unités

- Les **coordonnées géographiques** peuvent être affichées en degrés, minutes, secondes ou en degrés décimaux.
  - 1° d'angle = 60 minutes d'angle (1° = 60 ')
  - 1 minute d'angle = 60 secondes d'angle (1' = 60")
- Exemple : 31°14'44" => 31,245556°

  31 + 14/60 + 44/3600 = 31+0,2333...+0,01222...= 31,245555..
- Sur Internet, il y a plusieurs sites de **conversion** d'unités

  http://tool-online.com/conversion-angle.php

  https://www.coordonnees-gps.fr/conversion-coordonnees-gps

Exercice : exGeo1.html

- Trouvez sur Internet les coordonnées des lieux suivants (par exemple sur Wikipédia) :
  - Bruxelles – Grand'Place
  - Paris – Tour Eiffel
  - USA/Canada – Chutes de Niagara
  - Inde – Taj Mahal
  - Pérou – Cité Inca de Machu Picchu
- Créez une page **exGeo1.html** pour afficher les données trouvées sous forme d'un **tableau** (Pays – Lieu/Site – Longitude – Latitude)

Comment obtenir les coordonnées

- Différentes techniques sont utilisées 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** (mobiles)
  - Par triangulation **GSM** (mobiles 3G/4G/5G)
  - Par triangulation **WiFi** (mobiles et bases de données - adresses MAC)
  - Par **adresse IP** (correspondance avec des bases de données)
  - etc

Comment savoir si le navigateur supporte la géolocalisation ?

- On doit tester l'objet **geolocation** (réf : MDN), membre de l'objet **navigator** (réf : MDN) avec **Javascript**.
- L'objet navigator.geolocation renvoie true si le navigateur supporte et false sinon

```
if (navigator.geolocation)
   {
    // true – Le navigateur supporte    }
else
  {
    // false – Le navigateur ne supporte pas }
```

Exercice : exGeo2.html

- Créez une page **exGeo2.html** pour tester si le navigateur supporte la géolocalisation
- Ecrivez une fonction en JavaScript : **detecterGeoLoc()** qui affiche dans un paragraphe si votre navigateur la supporte ou non
- **Appelez cette fonction** dans la page exGeo2.html en appuyant sur un **bouton**
- **Testez** cette page dans les navigateurs déjà installés (Chrome, Firefox, Safari, Edge, Opera,...)

Consentement de l'utilisateur

- HTML5 peut récupérer les coordonnées de l'utilisateur **uniquement** après que celui-ci lui ait donné son **consentement**
- Dès qu'on fait appel aux fonctions suivantes :
  - **getCurrentPosition**
  - **watchPosition**

  Le navigateur affiche un **message d'alerte** demandant **l'autorisation**
- En principe, ceci ne fonctionne que si la page est placée sur un **serveur** (localhost ou sur Internet)

Types de positionnement

- On distingue **deux types de positionnement** avec la géolocalisation :
  - le **positionnement ponctuel** : c'est-à-dire une position fixe à un moment donné
  - le **suivi continu** : c'est-à-dire la répétition de la recherche du positionnement pour un suivi à intervalle régulier
- On utilise alors une fonction pour chaque cas :
  - **getCurrentPosition** dans le premier cas
  - **watchPosition** dans le second cas

Positionnement ponctuel

- On utilise la méthode **getCurrentPosition** de l'objet geolocation, membre de l'objet navigator, avec Javascript
- Cette méthode utilise **3 paramètres** :
  - une **fonction de callback** pour récupérer la position
  - une **fonction de callback** pour récupérer l'erreur
  - des **options**
- La détection pouvant nécessiter un temps variable (et incertain), les **appels** aux méthodes de géolocalisation sont **asynchrones**, elles rendent la main pour la suite du code.

getCurrentPosition

- Il faut créer deux fonctions de callback :
  - **positionTrouvee()** : pour récupérer les données de positionnement si le consentement est donné
  - **erreurPosition()** : pour indiquer les erreurs survenues pendant la géolocalisation
- On recherche la position avec JavaScript :

```
navigator.geolocation.getCurrentPosition(positionTrouvee,erreurPosition);
```

L'objet de gestion de position

- La fonction de callback qui gère la position récupère en paramètre un objet **position** qui a des propriétés et méthodes.
- Cet objet a comme propriété l'objet **coords** qui contient les coordonnées géographiques :
  - **position.coords.latitude** permet de récupérer la latitude exprimée en degrés
  - **position.coords.longitude** permet de récupérer la longitude exprimée en degrés
  - **position.coords.altitude** permet de récupérer la longitude exprimée en mètres

Exemple

```
function positionTrouvee(position) {
document.getElementById("lat").innerHTML =
position.coords.latitude + " °";
document.getElementById("long").innerHTML =
position.coords.longitude + " °";
document.getElementById("alti").innerHTML =
position.coords.altitude + " m";
}
```

L'objet de gestion d'erreurs

- La fonction de callback qui gère les erreurs récupère en paramètre un objet **error** qui a des propriétés et méthodes.
- Cet objet a comme propriété l'objet **error.code** qui renvoie un code d'erreur :
  - error.PERMISSION_DENIED si l'utilisateur n'a pas donné son consentement pour sa position
  - error.POSITION_UNAVAILABLE si la position n'a pas pu être trouvée ou calculée
  - error.TIMEOUT si le délai de réponse est dépassé
  - error.UNKNOWN_ERROR pour tout autre erreur

Exemple

```
function erreurPosition(erreur) {
    switch(erreur.code) {
        case erreur.PERMISSION_DENIED:
        alert("PERMISSION_DENIED : l'utilisateur n'a pas autorisé l'accès à sa position."); break;
        case erreur.POSITION_UNAVAILABLE:
        alert("POSITION_UNAVAILABLE : la position n'a pas pu être déterminée."); break;
        case erreur.TIMEOUT:
        alert("TIMEOUT : le service n'a pas répondu à temps."); break;
        case erreur.UNKNOWN_ERROR:
        alert("UNKNOWN_ERROR : une erreur inconnue s'est produite");}
}
```

Exercice : exGeo3.html

- Créez une page **exGeo3.html** pour récupérer la position de l'utilisateur
- Ecrivez une fonction en JavaScript : **rechercheP**osition() qui affiche soit la position courante si elle peut être détectée, soit un message d'erreur si la géolocalisation n'est pas possible (avec les 2 fonctions de callback vues plus haut)
- **Appelez cette fonction** dans la page exGeo3.html
- **Testez** cette page dans différents navigateurs (Chrome, Firefox, Safari, Edge, Opera,...)

Autres informations détaillées

- L'objet **coords** permet de récupérer d'autres informations mais pour cela il faut que les **options** soient définies.
  - **position.coords.accuracy** donne la précision de la mesure de la position, exprimée en mètres
  - **position.coords.altitudeAccuracy** donne la précision de la mesure de l'altitude, exprimée en mètres
  - **position.timestamp** donne l'heure de la mesure (à utiliser via un objet Date)
  - **position.coords.heading** donne la direction exprimée en degrés (position du Nord d'une boussole)
  - **position.coords.speed** donne la vitesse de déplacement exprimée en mètres par seconde

Options de getCurrentPosition

- Il y a des **options** pour cette méthode :
  - **enableHighAccuracy**:true (utilisation du GPS pour des coordonnées plus précises
  - **timeout**:10000 (délai de réponse en millisecondes, par défaut : Infinity)
  - **maximumAge**:0 (durée de vie d'une coordonnée)
- Pour les utiliser, on peut les appeler directement dans la méthode **getCurrentPosition** en les regroupant dans un **tableau statique** entre accolades (voir exemple suivant)

Exemple avec options

```
navigator.geolocation.getCurrentPosition(
  positionTrouvee,
  erreurPosition,
  { enableHighAccuracy:true,
  timeout:10000,
  maximumAge:0 }
);
```

- positionTrouvee est la fonction de callback qui récupère la position (coordonnées géographiques)
- erreurPosition est la fonction de callback qui récupère les éventuels messages d'erreur

Exercice : exGeo4.html

- Créez une page **exGeo4.html** pour récupérer la position de l'utilisateur (avec les informations détaillées)
- Ecrivez une fonction en JavaScript : **rechercheCompletePosition()** qui affiche soit la position courante si elle peut être détectée, soit un message d'erreur si la géolocalisation n'est pas possible (complétez l'exercice précédent)
- **Appelez cette fonction** dans la page exGeo4.html
- **Testez** cette page dans différents navigateurs (Chrome, Firefox, Safari, Edge, Opera,...)

Suite de l'exercice 4

- On devrait idéalement tester dans les différents navigateurs mais aussi dans différents terminaux (PC desktop, GSM/Smartphone, avec ou sans GPS, avec différentes valeurs pour les options) pour constater les variations de précision

Suivi continu

- Avec HTML5, il est aussi possible de faire un **suivi continu** du positionnement
- Il suffit d'utiliser la méthode **watchPosition()** à la place de getCurrentPosition. Les paramètres sont identiques

  id = navigator.geolocation.watchPosition(maPosition);
- Le résultat **id** retourné par cette méthode doit être **conservé dans une variable pour identifier le suivi**
- Pour stopper le suivi, on réutilise cette variable avec la méthode clearWatch()

  navigator.geolocation.clearWatch(id);

Exercice : exGeo5.html

- Créez une page exGeo5.html pour récupérer la position de l'utilisateur à intervalle régulier
- Ecrivez une fonction en JavaScript : **suivrePosition()** qui affiche soit la position courante si elle peut être détectée, soit un message d'erreur si la géolocalisation n'est pas possible
- **Appelez cette fonction** dans la page exGeo5.html
- Prévoyez **2 boutons** pour démarrer et stopper le suivi
- Testez cette page si possible uniquement sur un dispositif mobile en vous déplaçant (après le cours ;-)