

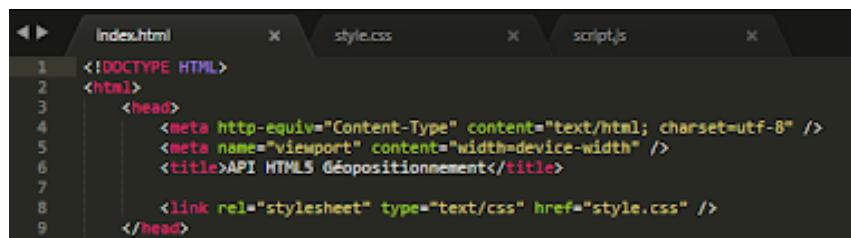
Rapport de Projet de Développement Web

Ce projet a pour but de créer une site web permettant d'afficher les coordonnées GPS (latitude, longitude) d'un utilisateur et d'afficher la distance le séparant de l'ESIREM à partir de ses coordonnées.

1. Création du fichier HTML

Dans ce fichier, nous définissons la structure de notre page. Pour commencer, nous définissons le DOCTYPE de notre fichier en tant que fichier HTML, nous créons la balise `<html>` dans laquelle nous allons définir la structure de notre page, à commencer par les balises `<head>` et `<body>`.

Dans la balise `<head>`, nous allons définir toutes les métadonnées de notre page, à commencer par le titre de celle-ci. Le titre sera “API HTML5 Géopositionnement”. Ensuite, le viewport sera défini par la largeur de l'appareil utilisé afin que l'utilisateur ne puisse pas zoomer sur notre page. Enfin, l'encodage des caractères de la page se fera en UTF-8. Nous allons également lier le futur fichier `.css` à notre fichier HTML dans la balise `<head>`.



```

1  <!DOCTYPE HTML>
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5          <meta name="viewport" content="width=device-width" />
6          <title>API HTML5 Géopositionnement</title>
7
8          <link rel="stylesheet" type="text/css" href="style.css" />
9      </head>

```

Déclaration de la balise `<head>`

Désormais, nous nous occupons de la balise `<body>` qui va être le corps de notre page et afficher les différents éléments lui appartenant. La balise `<h1>` permet d'afficher un titre, nous allons lui assigner le texte “Position et distance”. Ensuite nous déclarons deux éléments `<div>` qui auront leur utilité plus tard dans le fichier JavaScript notamment. Ces deux éléments `<div>` sont séparés par une ligne horizontale à l'aide de la balise `<hr>`. La première `<div>` sera associée à l'id position et la seconde, à l'id distance. A la fin de notre `<body>`, nous créons un lien vers le fichier JavaScript que nous allons définir juste après.



```

10
11      <body>
12          <h1 class="color"> Position et Distance </h1>
13
14          <div id="position" class="color">
15
16          </div>
17
18          <hr>
19
20          <div id="distance" class="color">
21          </div>
22
23          <script src="script.js"></script>
24
25      </body>
26

```

Déclaration de la balise `<body>`

2. Création du fichier JavaScript

Le lien entre le fichier HTML et le fichier JavaScript étant fait, nous allons commencer à créer les diverses fonctions permettant de calculer et d'afficher les distances et les coordonnées.

La fonction `geoSuccess()` permet d'afficher les coordonnées de l'utilisateur ainsi que la distance qui le sépare de l'ESIREM dans la `<div id="position">`. Pour cela, on utilise la géolocalisation du navigateur. La position de l'utilisateur est récupérée via la géolocalisation et la distance est calculée grâce à la fonction `calculDistance()` qui prendra en paramètre les coordonnées de l'utilisateur, les comparera avec celles de l'ESIREM et calculera la distance les séparant.

L'option position haute précision est définie dans la fonction `getLocation()` réutilisée dans `geoSuccess()` à l'aide de la commande `enableHighAccuracy:true`. Le timeout de `9000ms` est lui déclaré juste après.

Les coordonnées de l'utilisateur sont affichées avec 2 décimales à l'aide de la fonction `toFixed(2)`.

```

1  var p = document.getElementById("position");
2  var d = document.getElementById("distance");
3
4  function getLocation() {
5      if (navigator.geolocation) {
6          navigator.geolocation.getCurrentPosition(geoSuccess, geoError, {enableHighAccuracy:true, timeout:9000});
7      } else {
8          p.innerHTML = "Geolocation is not supported by this browser.";
9      }
10 }
11
12 function geoSuccess(position) {
13     var distance;
14     p.innerHTML = "Votre position est (" + position.coords.latitude.toFixed(2) +
15     ", " + position.coords.longitude.toFixed(2) + ") (avec une précision de 20m)"; //Affiche la latitude et de la longitude de l'utilisateur sur la page
16     distance = calculDistance(position.coords); //Calcul de la distance entre notre position et celle de l'ESIREM
17     d.innerHTML = "Vous êtes à " + distance.toFixed(2) + " km de l'ESIREM"; //Affichage de la distance sur la page
18 }

```

Fonctions getLocation() et geoSuccess()

La fonction `geoError()` permet de gérer les différentes erreurs pouvant survenir. Si la fonction `geoError()` est appelée, celle-ci va changer l'id des `<div>` de notre HTML. En effet, si l'id de la `<div>` était égal à "position" ou "distance", après l'appel de la fonction, celui-ci devient "erreur". De même, lorsque l'id est erreur, il devient "position" pour la première `<div>` et "distance" pour la deuxième après l'appel de la fonction. Ensuite, en fonction de la source de l'erreur, le navigateur affichera un message correspondant à l'erreur survenue afin que l'utilisateur comprenne le problème.

```

50
51     function geoError(error){
52         if(p.hasAttribute('id','position')){ //si l'ID est "position" alors change l'ID en "erreur"
53             p.removeAttribute('id','position');
54             p.setAttribute('id','erreur');
55         }
56         else{ //si l'ID est "erreur" alors change l'ID en "position"
57             p.removeAttribute('id','erreur');
58             p.setAttribute('id','position');
59         }
60         if(d.hasAttribute('id','distance')){ //si l'ID est "distance" alors change l'ID en "erreur"
61             d.removeAttribute('id','distance');
62             d.setAttribute('id','erreur');
63         }
64         else{ //si l'ID est "erreur" alors change l'ID en "distance"
65             d.removeAttribute('id','erreur');
66             d.setAttribute('id','distance');
67         }
68     }
69
70     switch(error.code) {
71         case error.PERMISSION_DENIED: //si l'utilisateur n'a pas autorisé la géolocalisation
72             p.innerHTML = "User denied the request for Geolocation."
73             break;
74         case error.POSITION_UNAVAILABLE: //si la position est introuvable
75             p.innerHTML = "Location information is unavailable."
76             break;
77         case error.TIMEOUT: //si le temps d'attente est trop long (>9000ms)
78             p.innerHTML = "The request to get user location timed out."
79             break;
80         case error.UNKNOWN_ERROR: //si l'erreur est inconnue
81             p.innerHTML = "An unknown error occurred."
82             break;
83     }
84 }
85

```

Fonctions geoError()

Ensuite, il nous est demandé d'implémenter la fonction `calculDistance()` qui prendra en paramètres les coordonnées de l'utilisateur. La deuxième fonction à déclarer est la fonction `degressesEnRadians()`. Celle-ci prend en paramètre une valeur en degrés et la convertit en radians afin que le calcul se fasse dans les bonnes dimensions.

Il nous est demandé ensuite de modifier cette fonction afin de mettre comme coordonnées de destinations celles de l'ESIREM. Or après avoir testé notre site, nous avons remarqué que le calcul de la distance n'était pas juste malgré une certaine imprécision. Nous avons donc vérifié les coordonnées de l'ESIREM et elles ne correspondent pas à celles données dans l'énoncé. Nous avons donc cherché les coordonnées de notre école afin d'avoir une distance plus réaliste (les coordonnées données nous emmenaient à Fontaine d'Ouche).

```

55     function calculDistance(startCoords) {
56         var startLatRads = degreesEnRadians(startCoords.latitude); //Conversion en radians de la latitude de l'utilisateur
57         var startLongRads = degreesEnRadians(startCoords.longitude); //Conversion en radians de la longitude de l'utilisateur
58         var destLatRads = degreesEnRadians(47.3121843); //Conversion en radians de la latitude de l'ESIREM
59         var destLongRads = degreesEnRadians(5.0736829); //Conversion en radians de la longitude de l'ESIREM
60         var Radius = 6371; // rayon de la Terre en km
61         var distance = Math.acos(Math.sin(startLatRads) * Math.sin(destLatRads) +
62             Math.cos(startLatRads) * Math.cos(destLatRads) *
63             Math.cos(startLongRads - destLongRads)) * Radius;
64         return distance;
65     }

```

Position et Distance

Votre position est {47.31, 5.06} (avec une précision de 20m)

Vous êtes à 0.84 km de l'ESIREM

Fonction calculDistance() et aperçu page Web

3. Création du fichier CSS

Le lien entre le fichier HTML et le fichier CSS étant déjà réalisé, nous allons désormais définir les différents styles de notre page afin de la rendre plus propre et plus attrayante.

Une contrainte nous est imposée pour les écrans de moins de 600px de large. Nous allons donc créer deux fonctions : une qui sera appelée pour les fenêtres de largeur maximum 600px et une autre pour les fenêtres d'au moins de 600px. Commençons par définir le cas où la fenêtre fait moins de 600px de large.

```

1  /* Fonctions correspondant aux fenêtres de moins de 600px de large */
2  @media screen and ( max-width: 600px ) {
3      body{
4          text-align: left;          /* Allignement du texte: à gauche */
5          width: 90%;              /* Largeur de la police: à 90% */
6          margin-left: auto;
7          margin-right: auto;       /* Fenêtre: centrée */
8          border: 3px dashed #d87093; /* Bordure: largeur 3px, de pointillés, rose */
9      }
10
11     .color{
12         background-color: #d87093; /* Couleur d'arrière-plan: rose */
13     }
14
15     h1{
16         color: white;           /* Police de <h1>: blanche */
17     }
18 }

```

CSS pour page < 600px et aperçu page Web

Ensuite, nous devons définir le style des fenêtres de largeur supérieure à 600px.

```

23  /* Fonctions correspondant aux fenêtres d'au moins 600px de large */
24  @media screen and ( min-width: 601px ){
25      body{
26          text-align: center;        /* Allignement du texte: centré */
27          width: 50%;              /* Largeur de la police: à 50% */
28          border: 3px solid #000000; /* Bordure: largeur 3px, pleine, noire */
29          margin-left: auto;
30          margin-right: auto;      /* Fenêtre: centrée */
31      }
32
33      /* Si id de div = erreur */
34      #erreur{
35          font-size: 16pt;           /* Taille de la police: 16pt */
36          background-color: red;    /* Couleur d'arrière-plan: rouge */
37      }
38
39      /* Si id de div = position ou distance */
40      #position, #distance{
41          font-size: 16pt;           /* Taille de la police: 16pt */
42          max-width: 100%;          /* Largeur de la police: à 100% */
43          padding-top: 50px;         /* Ecart du haut de la fenêtre: 50px */
44          padding-bottom: 50px;        /* Ecart du bas de la fenêtre: 50px */
45          text-align: center;        /* Allignement du texte: centré */
46          background-color: #f0ffff;  /* Couleur d'arrière-plan: bleu-blanc */
47  }
48 }

```

CSS pour page > 600px et aperçu page Web

Conclusion :

Pour conclure, même si l'utilisation du langage HTML et du langage CSS reste élémentaire, le Javascript est pour nous un outil extrêmement utile qui élargit le champ des possibilités pour améliorer et compléter l'utilisation de l'HTML et CSS. La partie la plus complexe est certainement le Javascript, néanmoins ce TP nous a permis d'en apprendre plus sur ce langage de programmation et a réussi à nous initier au développement web dans son ensemble.