Eskimon
Auteur externe

Google Maps JavaScript API V3

12 août 2019

Table des matières

Dans le cadre de l'unité d'enseignement PRO (Projet) de la HEIG-VD (Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud), ce tutoriel a pour but de fournir une introduction à l'API Google Maps V3, qui à l'heure actuelle (31.05.10) vient de perdre sa dénomination "labs" ("labs" étant la dénomination Google pour "version béta").

Pour pouvoir suivre ce tutoriel, il faut avoir un minimum de connaissances en Javascript et XHTML .

Voici les points qui sont abordés dans ce tutoriel :

— Affichage d'une carte Google Maps
— Création d'overlays (marqueurs, polygones et polylignes)
— Gestion des événements souris

Ce tutoriel a légèrement été repris par Eskimon pour revoir quelques exemples, remettre au gout du jour quelques aspects et ajouter des exemples *live*

1. Affichage d'une carte Google Maps

Voici le code minimal pour créer une page HTML affichant une carte.

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <!--
            Elément Google Maps indiquant que la carte doit être affiché
            en plein écran et qu'elle ne peut pas
            être redimensionnée par l'utilisateur
        -->
        <meta name="viewport"
            content="initial-scale=1.0, user-scalable=no" />
        <!--
            Inclusion de l'API Google MAPS
            Le paramètre "sensor" indique si cette application utilise
            un capteur pour déterminer la position de l'utilisateur (smartphone)
            Cet include pourrait aussi etre fait a la fin du <body>
        -->
        <script type="text/javascript"
            src="http://maps.google.com/maps/api/js?sensor=false"></script>
        <!-- Une feuille de style éventuel -->
        <link rel="stylesheet" href="style.css">

        <title>Tutoriel Google Maps</title>
    </head>
    <body>
        <!-- Le conteneur de notre carte -->
        <div id="carte" style="width:400px; height:500px"></div>
        <!-- Le script qui va créer notre carte -->
        <script type="text/javascript">
            function initialiser() {
                // Objet representant une coordonnée
                var latlng = new google.maps.LatLng(46.71109,
                    1.7191036);
                /*
                    Objet contenant des propriétés avec des identificateurs
                    prédéfinis dans Google Maps permettant de définir des
                    options d'affichage de notre carte
                */
                var options = {
                    center: latlng,
                    zoom: 5,
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                };

```

```
42                  /*
43                      Constructeur de la carte qui prend en paramêtre
    le conteneur HTML
44                      dans lequel la carte doit s'afficher et les
    options
45                  */
46                  var carte = new
                        google.maps.Map(document.getElementById("carte"),
                        options);
47              }
48
49              // On lance l'initialisation de notre carte
50              initialiser();
51          </script>
52      </body>
53  </html>
```

Les 3 éléments importants que l'on retrouve dans ce code sont

1. l'importation de la librairie Google Maps grâce aux balises `<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />` et `<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>` ;
2. le positionnement dans le corps de la page HTML d'une balise `<div id="...">` avec un certain id (identificateur que l'on utilisera dans le code Javascript lors de l'instanciation de la carte Google Maps pour définir son emplacement) et enfin ;
3. l'instanciation d'un objet de classe `google.maps.Map` représentant la carte qui sera affichée.

1.1. Options de la carte

Pour créer une carte, il faut nécessairement définir les options suivantes, comme nous avons pu le voir dans le code ci-dessus :

— `center` : centre de la carte
— `zoom` : agrandissement de la carte
— `mapTypeId` : type de la carte

Nom	Signification	Valeurs
`center`	**centre de la carte**	**coordonnées en latitude et longitude**
`zoom`	agrandissement de la carte	`0` à `20` (bornes comprises)

`mapTypeId`	type de la carte	`google.maps.MapTypeId.ROADMAP`, `google.maps.MapTypeId.SATELLITE`, `google.maps.MapTypeId.HYBRID`, `google.maps.MapTypeId.TERRAIN`

Table :Propriétés

Dans Google Maps les coordonnées sont sous formes de latitude et de longitude. La librairie Google Maps met à disposition un constructeur permettant de créer un objet de classe `google.maps.LatLng(lat:number, lng:number)` qui prend en paramètre des nombres représentant, respectivement, la latitude et la longitude.

Voici d'autres options intéressantes que l'on peut spécifier pour une carte :

- `disableDoubleClickZoom` : Si sa valeur est mise à `true`, cela désactive l'agrandissement en faisant un double-clic avec la souris.
- `draggable` : Si sa valeur est mise à `true`, cela désactive le fait de pouvoir faire glisser la carte en maintenant le clic sur elle.
- `scrollwheel` : Si sa valeur est mise à `true`, cela désactive l'agrandissement avec le scroll de la souris.

Par défaut, ces trois options ont la valeur `true`.

Avec le code précédent, vous devriez obtenir un rendu similaire a celui-ci :

!(https ://jsfiddle.net/8bbe9pkx/1/)

2. Création d'overlays

Maintenant que l'on sait comment afficher une carte, il se peut que l'on ait besoin de disposer des éléments sur elle, pour, par exemple, indiquer un certain emplacement ou encore définir une zone géographique. Ceci peut se faire à l'aide des *overlays*.

Les *overlays* sont des éléments graphiques que l'ont peut poser ou dessiner sur une carte Google Maps.

Les *overlays* auxquels on s'intéresse ici sont les suivants :

- les marqueurs qui permettent d'indiquer un point sur la carte à la manière d'un drapeau
- les *polylines* qui permettent, par exemple, de faire des tracés de vols d'avion
- les polygones qui permettent de dessiner sur la carte une zone géographique

2.1. Les marqueurs

Les marqueurs permettent de situer un point précis sur une carte.

Pour créer et afficher un marqueur, il faut au minimum spécifier une position (en latitude et longitude avec le constructeur `google.maps.LatLng()` vu dans la partie précédente "Affichage d'une carte Google Maps") et la carte sur laquelle le marqueur doit être affiché.

```
1  function initialiser() {
2      var latlng = new google.maps.LatLng(46.779231, 6.659431);
3      var options = {
4          center: latlng,
5          zoom: 5,
6          mapTypeId: google.maps.MapTypeId.ROADMAP
7      };
8      var carte = new
           google.maps.Map(document.getElementById("carte"), options);
9
10     /***************Nouveau code***************/
11
12     //création et placement du marqueur
13     var marqueur = new google.maps.Marker({
14         position: new google.maps.LatLng(44.1264415, 4.8036921),
15     map: carte // la variable js représentant la carte
16     });
17
18     /******************************************/
19 }
```

i Utilisez le squelette HTML vu dans la partie précédente "Affichage d'une carte Google Maps" et remplacez le code de la fonction Javascript `initialiser()` qui s'y trouve par le code ci-dessus. Faites de même pour tous les prochains exemples.

Ceci affiche un marqueur rouge que l'on voit sur la carte.

!(https ://jsfiddle.net/8bbe9pkx/2/)

Il est possible de rendre le marqueur *draggable* (c'est-à-dire permettre qu'on puisse le déplacer au moyen de la souris par un glisser-déposer). Pour ce faire, on peut soit, lors de la création du marqueur, spécifier dans les options `draggable: true` (entre les accolades dans les paramètres du constructeur, ici `new google.maps.Marker({...})`), soit, après avoir créé le marqueur `marqueur`, faire un `setDraggable(true)`. Essayez dans l'exemple ci-dessus !

De la même manière, on peut aussi modifier l'image du marqueur par la propriété `icon: "./mon_image.png"` ou la fonction `setIcon("./mon_image.png")` en pointant sur une image placée dans le bon dossier.

2.2. Les polylines

Les *polylines* permettent de dessiner des lignes droites attachées les unes aux autres sur la carte. Ceci peut permettre, par exemple, de dessiner un itinéraire sur la carte Google Maps.

Pour créer et dessiner un *polyline*, il est nécessaire de définir son chemin, c'est-à-dire les coordonnées par lesquelles il passe. Pour ce faire, il faut créer un tableau dont les éléments sont des instances de la classe `google.maps.LatLng()`.

Voici le tableau du tour de france que Clem va faire quand elle va rendre visite aux zesteurs :

```
// chemin du tracé du futur polyline
var tourdefrance = [
    new google.maps.LatLng(48.858859  ,  2.3470599), // Paris
    new google.maps.LatLng(45.7579555 ,  4.8351209), // Lyon
    new google.maps.LatLng(47.238222  , -1.5609655), // Nantes
    new google.maps.LatLng(48.1159156 , -1.6884545), // Rennes
    new google.maps.LatLng(50.6310675 ,  3.0471604), // Lille
    new google.maps.LatLng(48.858859  ,  2.3470599)  // Paris
];
```

Lorsque ceci est fait, il faut créer le *polyline* en spécifiant pour la propriété `path` le tableau déclaré ci-dessus.

```
var traceTdF = new google.maps.Polyline({
    path: tourdefrance,     // chemin du tracé
    strokeColor: "#d35400", // couleur du tracé
    strokeOpacity: 1.0,     // opacité du tracé
    strokeWeight: 2         // grosseur du tracé
});
```

Enfin, afin d'afficher le tracé sur la carte, il faut utiliser la méthode `setMap()` de notre objet `traceTdF`.

```
//lier le tracé à la carte
//ceci permet au tracé d'être affiché sur la carte
traceTdF.setMap(carte);
```

i Au lieu de faire un `setMap()`, on aurait pu, lors de la création du *polyline*, spécifier la propriété `map` dans les paramètres du constructeur `new google.maps.Polyline({...})`.

Quand on ajoute une propriété dans les paramètres du constructeur, il ne faut pas oublier de vérifier que seule la dernière définition de propriété ne se termine pas par une virgule et qu'à la fin de toutes les autres une virgule est présente.

Voici ce que le code de notre fonction `initialiser()` donne au final :

```
function initialiser() {
    var latlng = new google.maps.LatLng(46.779231, 6.659431);
    var options = {
        center: latlng,
        zoom: 5,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var carte = new
        google.maps.Map(document.getElementById("carte"), options);

    /*************** Nouveau code ***************/

    // redéfinition du centre de la carte
    carte.setCenter(new google.maps.LatLng(48.21109, 1.7191036));

    // redéfinition du zoom
    carte.setZoom(6);

    //chemin du tracé
    var tourdefrance = [
        new google.maps.LatLng(48.858859   ,  2.3470599), // Paris
        new google.maps.LatLng(45.7579555 ,  4.8351209), // Lyon
        new google.maps.LatLng(47.238222   , -1.5609655), // Nantes
        new google.maps.LatLng(48.1159156 , -1.6884545), // Rennes
        new google.maps.LatLng(50.6310675 ,  3.0471604), // Lille
        new google.maps.LatLng(48.858859   ,  2.3470599)  // Paris
    ];

    // creation de l'objet Polyline
    var traceTdF = new google.maps.Polyline({
        path: tourdefrance,     // chemin du tracé
        strokeColor: "#d35400", // couleur du tracé
        strokeOpacity: 1.0,     // opacité du tracé
        strokeWeight: 2         // grosseur du tracé
    });

    // lier le tracé (le polyline) à la carte
    // ceci permet au tracé d'être affiché sur la carte
    traceTdF.setMap(carte);

    /*******************************************/
```

```
41 }
```

Vous pouvez voir que j'ai ajouté sous le commentaire "Nouveau code" deux autres instructions. J'aurais pu mettre tout ceci dans les options de la carte, mais j'ai préféré faire de cette manière pour bien faire la différence entre le code original de notre fonction `initialiser()` et tout ce que nous venons d'ajouter.

Et voici le résultat :

!(https ://jsfiddle.net/8bbe9pkx/3/)

2.3. Les polygones

Sur une carte Google Maps, on peut aussi dessiner des polygones. Un exemple d'utilité pratique serait de définir par ce biais un secteur ou une zone.

Créer un polygone sur la carte est extrêmement similaire à la création d'un *polyline* que l'on vient de voir. Un peu comme avant, il faut :

1. Créer un tableau contenant tous les sommets du polygone
2. Créer le polygone avec le constructeur `google.maps.Polygon()`
3. Afficher le polygone sur la carte

Voici ce que donne le code :

```
1  function initialiser() {
2      var latlng = new google.maps.LatLng(46.779231, 6.659431);
3      var options = {
4          center: latlng,
5          zoom: 5,
6          mapTypeId: google.maps.MapTypeId.ROADMAP
7      };
8      var carte = new
           google.maps.Map(document.getElementById("carte"), options);
9
10     /*************** Nouveau code ***************/
11
12     //sommets du polygone
13     var tourdefrancePolygone = [
14         new google.maps.LatLng(48.858859  ,  2.3470599), // Paris
15         new google.maps.LatLng(45.7579555 ,  4.8351209), // Lyon
16         new google.maps.LatLng(47.238222  , -1.5609655), // Nantes
17         new google.maps.LatLng(48.1159156 , -1.6884545), // Rennes
18         new google.maps.LatLng(50.6310675 ,  3.0471604)  // Lille
```

```
19      ];
20
21      lePolygone = new google.maps.Polygon({
22          paths: tourdefrancePolygone, // sommets du polygone
23          strokeColor: "#d35400",      // couleur des bords du
                polygone
24          strokeOpacity: 0.8,          // opacité des bords du
                polygone
25          strokeWeight: 2,             // épaisseur des bords du
                polygone
26          fillColor: "##f39c12",       // couleur de remplissage du
                polygone
27          fillOpacity: 0.35            // opacité de remplissage du
                polygone
28      });
29
30      //lier le polygone à la carte
31      //ceci permet au polygone d'être affiché sur la carte
32      lePolygone.setMap(carte);
33
34      /*****************************************/
35  }
```

i Notez qu'il n'est pas nécessaire lors de la définition des sommets de la parcelle de définir le dernier sommet sur le premier afin de fermer le polygone ; le dernier sommet sera automatiquement relié au premier.

!(https ://jsfiddle.net/8bbe9pkx/5/)

3. Gestion des évènements souris

Il est possible avec Google Maps d'attacher des gestionnaires d'évènements à certains objets de Google Maps comme les cartes, les marqueurs, les polygones, etc.

Voici un tableau de quelques évènements que l'on peut gérer dans Google Maps :

Indentificateur de l'évènement	Généré quand on ...
`'click'`	**... clique avec la souris**
`'rightclick'`	... fait un clic-droit avec la souris
`'dblclick'`	... fait un double-clic avec la souris
`'drag'`	... déplace un objet au moyen de la souris par un glisser-déposer (généré plusieurs fois tout au long de cette action)

`'dragstart'`	... une fois tout au début d'un déplacement d'un objet au moyen de la souris par un glisser-déposer
`'dragend'`	... une fois tout à la fin d'un déplacement d'un objet au moyen de la souris par un glisser-déposer
`'mouseover'`	... lorsque le pointeur de la souris entre sur la surface d'un objet Google Maps
`'mouseout'`	... lorsque le pointeur de la souris sort de la surface d'un objet Google Maps

TABLE 3. – Evènements dans Google Maps

Créer un gestionnaire d'évènements se fait toujours de la même manière c'est pour cela qu'on ne fera que deux exemples : celui du `'click'` et celui du `'drag'`.
Voici ce à quoi le code de n'importe quel gestionnaire d'évènement ressemble :

```
1 //'evenement' est l'identificateur de l'évènement (voir tableau
      ci-dessus)
2 //obj est l'objet duquel nous souhaitons traiter les évènements
3 google.maps.event.addListener(obj,'evenement', function(event) {
4         /*code qui doit s'executer lors de l'evenement*/
5 });
```

i L'argument "event", dans le code ci-dessus, n'est utile que si l'on souhaite accèder à ses propriétés. Par exemple, la propiété latLng représentant une latitude et une longitude est disponiple pour un event de type MouseEvent passé par l'évènement clic.

3.1. 'click' (clic)

Reprenons le code que nous avions fait pour créer un marqueur dans la partie précédente "Création d'overlays" et ajoutons-y un gestionnaire d'évènements pour le clic sur le marqueur. Ce gestionnaire devra afficher un message d'alerte Javascript disant que le marqueur a été cliqué.
Voici à quoi le code ressemble :

```
1 function initialiser() {
2         var latlng = new google.maps.LatLng(46.779231, 6.659431);
3
4         var options = {
5                 center: latlng,
6                 zoom: 19,
7                 mapTypeId: google.maps.MapTypeId.ROADMAP
8         };
9
```

```
10          var carte = new
                google.maps.Map(document.getElementById("carte"),
                options);
11
12          //création du marqueur
13          var marqueur = new google.maps.Marker({
14                  position: new google.maps.LatLng(46.779231,
                        6.659431),
15                  map: carte
16          });
17
18          /***************Nouveau code***************/
19
20          //création du marqueur
21          var marqueur = new google.maps.Marker({
22                  position: new google.maps.LatLng(44.1264415,
                        4.8036921),
23                  map: carte
24          });
25
26          google.maps.event.addListener(marqueur, 'click', function()
                {
27                  alert("Le marqueur a été cliqué.");//message
                        d'alerte
28          });
29
30          /*****************************************/
31  }
```

!(https ://jsfiddle.net/8bbe9pkx/6/)

Avec les connaissances que nous avons à ce stade, nous pouvons écrire un code qui permette de créer des marqueurs dynamiquement sur la carte.
Reprenons de nouveau le code de la fonction `initialiser()` du début de ce tutoriel et ajoutons-lui un gestionnaire d'événement clic associé cette fois-ci à la carte et non à un marqueur, puis mettons-y un code permettant de créer un marqueur (à chaque clic, donc) avec la position `event.latLng` qui est une propriété de l'`event` de type `MouseEvent`.
Voici ce que donne le code :

```
1  function initialiser() {
2          var latlng = new google.maps.LatLng(46.779231, 6.659431);
3
4          var options = {
5                  center: latlng,
6                  zoom: 19,
7                  mapTypeId: google.maps.MapTypeId.ROADMAP
8          };
9
```

```
10          var carte = new
                google.maps.Map(document.getElementById("carte"),
                options);
11
12          /***************Nouveau code***************/
13
14          //tableau contenant tous les marqueurs que nous créerons
15          var tabMarqueurs = new Array();
16
17          //notez la présence de l'argument "event" entre les
                parenthèses de "function()"
18          google.maps.event.addListener(carte, 'click',
                function(event) {
19                  tabMarqueurs.push(new google.maps.Marker({
20                          position: event.latLng,//coordonnée de la
                                position du clic sur la carte
21                          map: carte//la carte sur laquelle le
                                marqueur doit être affiché
22                  }));
23          });
24
25          /*****************************************/
26  }
```

Si vous testez ce code, vous pourrez voir qu'un marqueur est créé lors de chaque clic de la souris sur la carte.

!(https ://jsfiddle.net/8bbe9pkx/7/)

3.2. 'dragend' (fin d'un glisser-déposer)

Pour illustrer la gestion de l'évènement `'dragend'`, nous utiliserons de nouveau le code que nous avons fait pour la création d'un marqueur dans la partie "Création d'overlays".
L'exemple qui suit affiche, lorsque l'on déplace le marqueur, un message d'alerte Javascript indiquant la nouvelle coordonnée du marqueur.

```
1  function initialiser() {
2          var latlng = new google.maps.LatLng(46.779231, 6.659431);
3
4          var options = {
5                  center: latlng,
6                  zoom: 19,
7                  mapTypeId: google.maps.MapTypeId.ROADMAP
8          };
9
10         var carte = new
               google.maps.Map(document.getElementById("carte"),
               options);
11
```

```
12          /***************Nouveau code***************/
13
14          //création du marqueur
15          var marqueur = new google.maps.Marker({
16                  position: new google.maps.LatLng(44.1264415,
                        4.8036921),
17                  map: carte
18          });
19
20          //ne pas oublier de rendre le marqueur "déplaçable"
21          marqueur.setDraggable(true);
22
23          google.maps.event.addListener(marqueur, 'dragend',
                function(event) {
24                  //message d'alerte affichant la nouvelle position
                        du marqueur
25                  alert("La nouvelle coordonnée du marqueur est : "+event.latLng)
26          });
27
28          /*****************************************/
29  }
```

!(https ://jsfiddle.net/8bbe9pkx/8/)

---

i

Ce tutoriel a pu être créé essentiellement grâce à la documentation mis en ligne par Google qui se trouve sur le lien suivant : https ://developers.google.com/maps/documentation/javascript/tutorial ?hl=FR

N'ayant pas pour but d'être exhaustif, ce tutoriel permet néanmoins la familiarisation avec l'API Google Maps V3.
Pour pouvoir aller plus loin dans l'utilisation de Google Maps, il y a la documentation officielle de Google , mais qui n'est malheureusement pas disponible en français.