

# Exemples d'applications de géolocalisation

## 1. Exemple 1 : Affichage de la carte de l'Ouest de la France

Dans ce premier exemple nous allons mettre en œuvre les fonctionnalités de base de l'API de géolocalisation nativement incluse dans HTML 5 et en conséquence prises en compte par les navigateurs récents (y compris sur les smartphones).

L'objectif va être très simple, afficher à l'écran un fond de carte Google Map présentant la région Ouest de la France (incluant donc la Bretagne) avec un centrage de la carte sur la ville de Rennes.

Dans la mesure où il s'agit de notre premier exemple de cartographie, le script HTML/JavaScript est reproduit ci-après dans son intégralité (ce ne sera pas systématiquement le cas pour les exemples suivants) :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!--
NOM DU SCRIPT : GEO_01.htm
REALISATION INFORMATIQUE : Christian VIGOUROUX
DATE DE CREATION : 01/01/2014
DATE DE DERNIERE MODIFICATION : 01/01/2014
OBJET : Gestion carte Google Map - Carte de l'Ouest de la France
(centrage sur Rennes)
-->

<!-- Début script HTML -->
<html>

  <!-- Début en-tête script HTML -->
  <head>

    <!-- Balise meta http-equiv & content -->
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <!-- Balise meta définissant la zone affichable -->
    <meta name="viewport" content="initial-scale=1.0, user-scalable=yes"/>
    <!-- NB : Le méta tag viewport précise au navigateur le comportement
    qu'il doit adopter pour afficher une page -->
    <!--      initial-scale=1.0 : Ouverture de la page
    avec une échelle à 100 % -->
    <!--      user-scalable=yes : Zoom possible
    de la part de l'utilisateur -->

    <!-- Feuille de styles CSS -->
    <style type="text/css">

      html, body, #maCarte
      {
        margin: 0;
        padding: 0;
        height: 80%;
      }
    </style>
  </head>
</html>
```

```

</style>

<!-- Titre du script HTML -->
<title>GEO_01</title>

<!-- Appel de l'API Google Map -->
<!-- NB : sensor=false signifie que l'application
ne propose pas de repérage GPS -->
<script type="text/Javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>

<!-- Script JavaScript de mise en place de la carte -->
<script type="text/Javascript">

    /* Fonction d'initialisation de la carte */
    function initialiserCarte()
    {

        /* Test pour savoir si le navigateur supporte l'API de
géolocalisation (W3C) */
        if (!navigator.geolocation)
        {
            /* Message d'alerte */
            alert("Votre navigateur ne gère pas
la géolocalisation");
        }
        /* valeur de retour */
        return false
    }

    /* Définition de la position de centrage de la carte
(centrée sur la ville de Rennes) */
    /* NB : Cf. http://code-postal.fr.mapawi.com/france/1/
1/arrondissement-de-rennes/3/84/rennes/35000/8412/ */
    var centreGoogleMap =
    new google.maps.LatLng(48.0833, -1.6833);

    /* Définition des options de la carte */
    var optionsGoogleMap =
    {
        /* Facteur de zoom */
        zoom: 8,
        /* Point de centrage */
        center: centreGoogleMap,
        /* Mode d'affichage de la carte (vue carte routière) */
        /* NB : google.maps.mapTypeId.ROADMAP ->
Affichage en mode Plan */
        /* google.maps.mapTypeId.SATELLITE ->
Affichage en mode Satellite */
        /* google.maps.mapTypeId.HYBRID ->
Affichage en mode Mixte (Plan/Satellite) */
        /* google.maps.mapTypeId.TERRAIN ->
Affichage en mode Relief */
        mapTypeId: google.maps.MapTypeId.ROADMAP
    }

```

```

    }

    /* Mise en place de la carte dans la division maCarte */
    var maCarte =
    new google.maps.Map(document.getElementById("maCarte"),
    optionsGoogleMap);

    }

</script>

</head>

<!-- Début section body du script HTML -->
<body onload="initialiserCarte()">

    <!-- Titre du traitement -->
    <h1>Editions ENI - JavaScript - GEO_01</h1>

    <!-- Début script JavaScript -->
    <script type="text/Javascript">

        /* Affichage du nom du script */
        alert("GEO_01");

    </script>

    <!-- Définition de la division dans laquelle la carte sera affichée -->
    <div id="maCarte" style="width:100%; height:100%"></div>

    <!-- Message à destination des internautes ayant un navigateur
    sans JavaScript -->
    <noscript>
        <p>Remarque importante :</p>
        <p>Pour utiliser une carte de type Google Map,
        il faut que JavaScript soit activé dans votre navigateur.</p>
    </noscript>

    <!-- Affichage du code source -->
    <br /><br /><br />
    <center>
    <a href="JavaScript:window.location='view-source:' + window.location">
        Code source
    </a>
    </center>

</body>

</html>

```

## **Section HTML <body>**

Débutons le commentaire par la section HTML <body>.

Cette section commence par le déclenchement de la fonction `initialiserCarte` :

```
<!-- Début section body du script HTML -->
<body onload="initialiserCarte()">
```

Ensuite une division identifiée par `maCarte` est définie. Elle recevra à terme la carte.

```
<!-- Définition de la division dans laquelle la carte sera affichée -->
<div id="maCarte" style="width:100%; height:100%"></div>
```

Ne nous attardons pas sur le reste de la section. Vous remarquerez simplement un jeu de balises `<noscript> ... </noscript>`, à peu près inutile désormais car tous les navigateurs supportent le JavaScript.

### **Section HTML <head>**

Cette section débute par la balise meta habituelle définissant le type de contenu et le jeu de caractères.

Une seconde balise meta spécifique à notre traitement est trouvée :

```
<!-- Balise meta définissant la zone affichable -->
<meta name="viewport" content="initial-scale=1.0, user-scalable=yes"/>
!-- NB : Le méta tag viewport précise au navigateur le comportement
qu'il doit adopter pour afficher une page -->
<!--      initial-scale=1.0 : Ouverture de la page
avec une échelle à 100 % -->
<!--      user-scalable=yes : Zoom possible
de la part de l'utilisateur -->
```

Cette balise indique au navigateur quel comportement il doit avoir lors de l'affichage de la page. Dans notre cas, la page est affichée à 100 % avec le zoom autorisé.

Ensuite une petite feuille de style CSS est intégrée, elle sert à définir la marge, le padding (marge supplémentaire entre le contenu et le bord) et la hauteur pour les éléments HTML `html`, `body` et `#maCarte`.

```
<!-- Feuille de styles CSS -->
<style type="text/css">

html, body, #maCarte
{
    margin: 0;
    padding: 0;
    height: 80%;
}
```




Ce livre ne fait pas d'apports sur les feuilles de style CSS bien que les exemples y fassent parfois un peu recours. Vous pourrez vous former efficacement aux CSS par la lecture des ouvrages "XHTML et CSS - Les nouveaux standards du code source [2ème édition]" de Luc VAN LANCKER ou de "HTML5, CSS3 et JavaScript - Développez vos


Étudions maintenant la fonction JavaScript `initialiserCarte`.

Avant il faut signaler l'utilisation de l'API Google comme suit :

```
<!-- Appel de l'API Google Map -->
<!-- NB : sensor=false signifie que l'application ne propose pas de
repérage GPS -->
<script type="text/Javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
```

Le paramètre `sensor="false"` précise que l'application n'utilise pas le repérage GPS (*Global Positioning System*).

 Source Wikipedia : le Global Positioning System (GPS) - que l'on peut traduire en français par "système de localisation mondial" - est un système de géolocalisation fonctionnant au niveau mondial. Il a été mis en place à l'origine par le Département de la Défense des États-Unis. Le GPS a connu un grand succès dans le domaine civil et a engendré un énorme développement commercial dans de nombreux domaines : navigation maritime, sur route, localisation de camions, randonnée, etc.

 Il est à noter que depuis la version 3 de l'API Google Map, il n'est plus nécessaire de demander à Google un code pour pouvoir utiliser une carte dans vos réalisations Web. Vous pouvez consulter les conditions d'utilisation à l'adresse suivante : <https://developers.google.com/maps/documentation/javascript/tutorial?hl=fr>. La gestion des clés Google sort du cadre de ce livre.

Bien que les navigateurs récents prennent tous en charge l'API de géolocalisation, il peut être intéressant de tester sa disponibilité :

```
/* Test pour savoir si le navigateur supporte l'API
de géolocalisation (W3C) */
if (!navigator.geolocation)
{
    /* Message d'alerte */
    alert("Votre navigateur ne gère pas la géolocalisation");
    /* Valeur de retour */
    return false
}
```

Passons ensuite à la définition de la position de centrage de la carte. Il a été décidé de la centrer autour de la ville de Rennes, qui a pour position 48.0833 en latitude et -1.6833 en longitude.

```
/* Définition de la position de centrage de la carte
(centrée sur la ville de Rennes) */
var centreGoogleMap = new google.maps.LatLng(48.0833, -1.6833);
```

De très nombreux sites donnent ces informations pour la grande majorité des villes françaises et mondiales.

Des compléments d'information (options) sont requis pour le bon affichage de la carte :

```

/* Définition des options de la carte */
var optionsGoogleMap =
{
    /* Facteur de zoom */
    zoom: 8,
    /* Point de centrage */
    center: centreGoogleMap,
    /* Mode d'affichage de la carte (vue carte routière) */
    mapTypeId: google.maps.MapTypeId.ROADMAP
}

```

Dans notre cas, un facteur de zoom de 8 a été choisi, la carte est bien évidemment centrée sur la position Rennes et l'affichage est en "mode routier" (ROADMAP).

D'autres modes d'affichage sont disponibles :

- google.maps.mapTypeId.SATELLITE : Mode Satellite
- google.maps.mapTypeId.HYBRID : Mode Mixte (Plan/Satellite)
- google.maps.mapTypeId.TERRAIN : Mode Relief

Terminons par l'affectation de la carte créée à la division HTML prévue pour son affichage :

```

/* Mise en place de la carte dans la division maCarte */
var maCarte = new google.maps.Map(document.getElementById("maCarte"),
optionsGoogleMap);

```

Nous retrouvons bien sûr dans cette instanciation, la division HTML `maCarte` et les options d'affichage `optionsGoogleMap`.

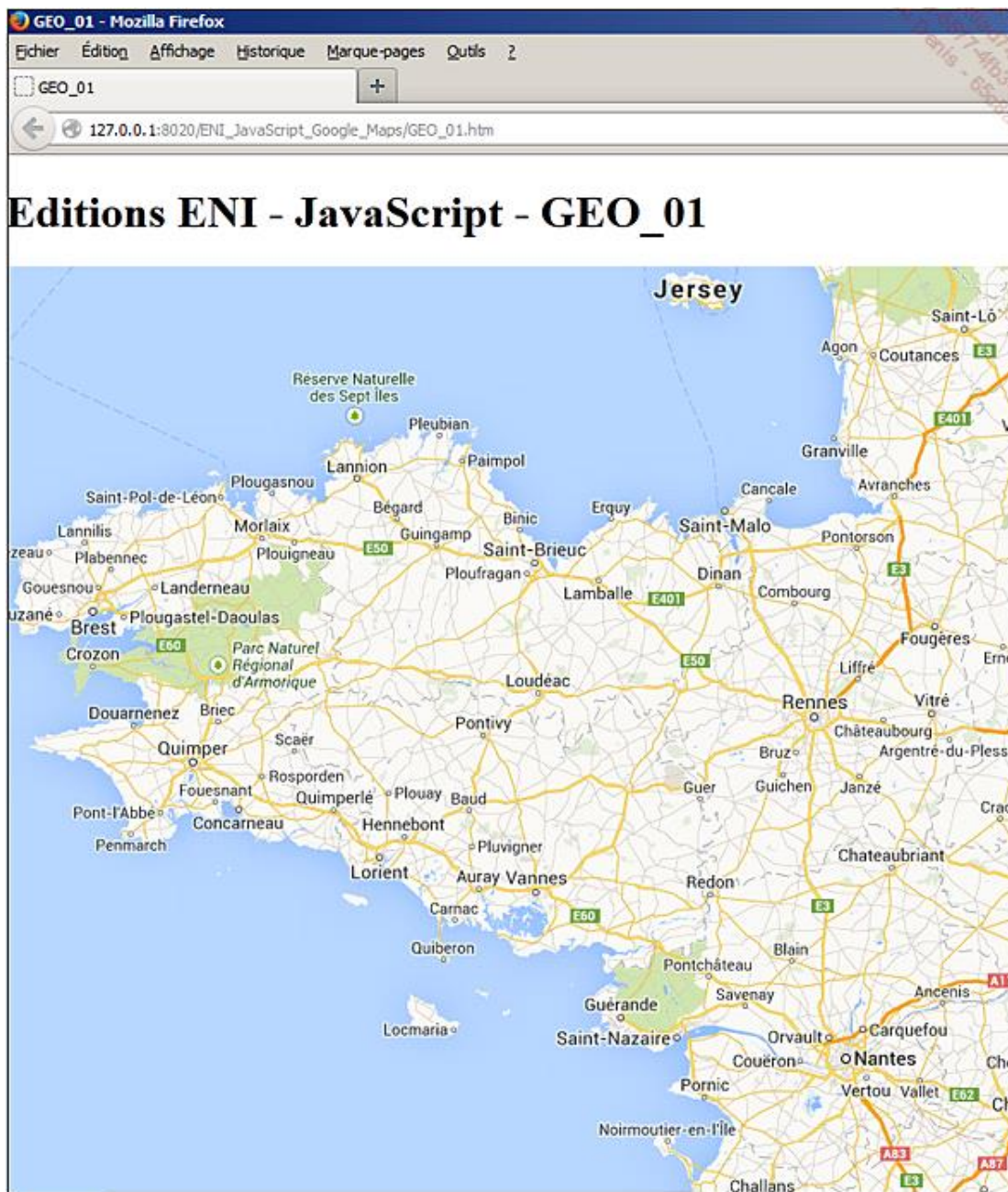
Il est à noter qu'avec ce script minimaliste, la carte s'affiche avec un certain nombre de dispositifs : zoom avant/arrière, décalage horizontal/vertical, icône de passage en mode Google Street View.



Source Google : Google Map avec Street View permet d'explorer les sites du monde entier en profitant d'images à 360 degrés des rues. Vous pouvez explorer des sites remarquables, découvrir des merveilles naturelles, suivre une route touristique, entrer dans des restaurants...

### **Compte rendu d'exécution**

L'exécution du script `GEO_01.htm` génère cet affichage :



➤ La carte ne semble pas ici centrée sur la position de la ville de Rennes, en réalité elle l'est dans le navigateur.

## 2. Exemple 2 : Affichage de la carte de l'Ouest de la France (marqueur)

Ce deuxième exemple va être assez proche du précédent. Sa plus-value va être le placement sur la position géographique de Rennes d'un marqueur.

Les explications de base sur la mise en place de la carte ne seront par fournies une deuxième fois.

Une marque est une icône (il est possible d'en modifier l'image de base) sur laquelle l'utilisateur pourra cliquer pour obtenir une information contextuelle (sous forme d'une fenêtre de type pop-up).

### Section HTML <body>

Dans la section HTML `<body>`, nous trouverons la division d'affichage `maCarte` déjà rencontrée dans l'exemple 1.

Par contre pour ce qui est du déclenchement de la fonction `initialiserCarte`, nous procéderons différemment, par l'intermédiaire du gestionnaire d'événements de la Google Map. Nous y viendrons plus tard.

## **Section HTML `<head>`**

Les balises meta et la feuille de style CSS intégrée sont identiques à celles vues dans l'exemple 1.

Vous retrouverez aussi l'appel à l'API Google Map à l'identique.

Passons à l'étude de la fonction `initialiserCarte`.

La définition de la position de centrage, les options et la mise en place de la carte dans la division HTML `maCarte` ne changent pas par rapport à l'exemple précédent.

Il reste à voir la gestion du marqueur. Le code intégral de cette partie est reproduit ci-après :

```
/* Mise en place d'un marqueur pour repérer la ville de Rennes */
/* NB1 : position définit la position en latitude/longitude du marqueur
   map permet l'affectation du marqueur à la carte */
/* NB2 : La mise en place du marqueur doit suivre l'affichage
   de la carte dans la division (et non pas l'inverse) */
var marqueurRennes = new google.maps.Marker({
  position: new google.maps.LatLng(48.0833, -1.6833),
  map: maCarte,
  title: "Rennes, principale ville de Bretagne"
});

/* Texte explicatif pour la ville de Rennes */
var commentairesRennes =
  "<div>" +
  "<h1>Rennes</h1>" +
  "Rennes est une commune française, chef-lieu du département  

  d'Ille-et-Vilaine<br />" +
  "et de la région Bretagne, ainsi que l'une des capitales historiques  

  du duché de Bretagne.<br />" +
  "Cette ville se situe à l'est de la Bretagne à la confluence  

  de l'Ille et de la Vilaine.<br />" +
  "Ses habitants sont appelés les Rennais et les Rennaises" +
  "<br />Site <a href='http://metropole.rennes.fr/'>Rennes Métropole</a>" +
  "</div>";

/* Constructeur de la fenêtre explicative associée à la ville de Rennes */
var fenetreRennes = new google.maps.InfoWindow({
  content: commentairesRennes
});

/* Affichage d'une fenêtre explicative sur clic
sur le marqueur marqueurRennes */
google.maps.event.addListener(marqueurRennes, "click", function() {
  fenetreRennes.open(maCarte, marqueurRennes);
});
```



Commençons par la mise en place du marqueur repérant la ville de Rennes :

```
/* Mise en place d'un marqueur pour repérer la ville de Rennes */
/* NB1 : position définit la position en latitude/longitude du marqueur */
/*      map permet l'affectation du marqueur à la carte */
/* NB2 : La mise en place du marqueur doit suivre l'affichage
de la carte dans la division (et non pas l'inverse) */
var marqueurRennes = new google.maps.Marker({
    position : new google.maps.LatLng(48.0833, -1.6833),
    map: maCarte,
    title: "Rennes, principale ville de Bretagne"
});
```

La marque est placée par l'intermédiaire de l'objet Marker de l'API Google Map.

Son constructeur a pour paramètre un tableau d'options :

- position : position de la marque sur la carte,
- map : carte sur laquelle est affichée la marque,
- title : texte affiché au survol de la souris.

Ensuite un texte explicatif est prévu pour un affichage ultérieur dans une fenêtre pop-up :

```
/* Texte explicatif pour la ville de Rennes */
var commentairesRennes =
    "<div>" +
    "<h1>Rennes</h1>" +
    "Rennes est une commune française, chef-lieu du département
d'Ille-et-Vilaine<br />" +
    "et de la région Bretagne, ainsi que l'une des capitales historiques
du duché de Bretagne.<br />" +
    "Cette ville se situe à l'est de la Bretagne à la confluence de
l'Ille et de la Vilaine.<br />" +
    "Ses habitants sont appelés les Rennais et les Rennaises" +
    "<br />Site <a href='http://metropole.rennes.fr/'>Rennes Métropole</a>" +
    "</div>";
```

Ce texte est ensuite affecté à une fenêtre (nommée fenetreRennes) ayant pour constructeur :

```
/* Constructeur de la fenêtre explicative associée à la ville de Rennes */
var fenetreRennes = new google.maps.InfoWindow({
    content: commentairesRennes
});
```

Le constructeur de l'objet InfoWindow a pour paramètre le commentaire stocké dans la variable commentairesRennes.


Une fois la fenêtre définie, il faut en programmer l’affichage, qui ne sera effectif que si l’opérateur clique sur le marqueur. Ceci est géré par le gestionnaire d’événement de la Google Map :

```
/* Affichage de la fenêtre explicative sur clic
sur le marqueur marqueurRennes */
google.maps.event.addListener(marqueurRennes, "click", function() {
    fenetreRennes.open(maCarte, marqueurRennes);
});
```

Sur l’événement `click` sur le marqueur, l’affichage de la fenêtre `fenetreRennes` est effectué et ceci dans la division HTML `maCarte`.

Dernier point concernant cet exemple, il reste à prévoir l’appel de la fonction `initialiserCarte` :

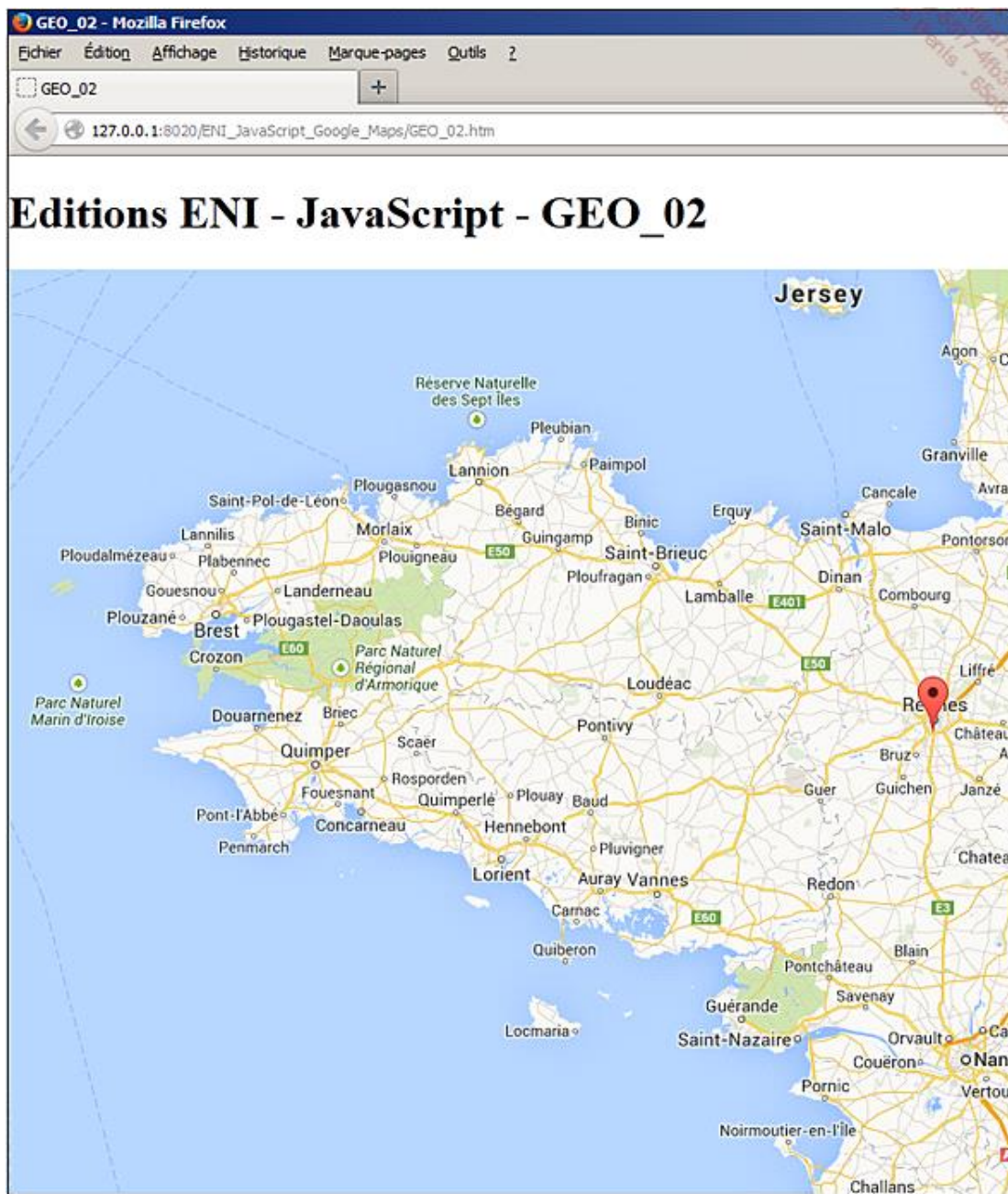
```
/* Appel de la fonction initialiserCarte par le gestionnaire d’événements */
google.maps.event.addDomListener(window, "load", initialiserCarte)
```

 Attention : cet appel événementiel s’effectue dans la section HTML `<head>` après la description de la fonction `initialiserCarte`.

Une alternative a été vue dans l’exemple 1 : `<body onload="initialiserCarte()">`

### **Compte rendu d’exécution**

L’exécution du script `GEO_02.htm` affiche ceci :



### 3. Exemple 3 : Affichage de la carte de l'Ouest de la France (marqueur et cercles de population)

Dans ce nouvel exemple, nous reprendrons la trame des deux exemples précédents. L'apport va être la mise en place de cercles dont le rayon sera proportionnel à la population vivant dans les quatre préfectures de la Bretagne. Seule cette partie sera commentée ici.

#### Section HTML <body>

Aucune nouveauté n'est à signaler par rapport à l'exemple 2.

#### Section HTML <head>

Dans la fonction `initialiserCarte`, outre le positionnement d'un marqueur sur la ville de Rennes, un second

marqueur est placé sur Quimper. Ce marqueur est personnalisé par un changement d'icône.

```
/* Mise en place d'un marqueur pour repérer la ville de Quimper */
/* NB1 : position définit la position en latitude/longitude du marqueur */
/*      map permet l'affectation du marqueur à la carte */
/* NB2 : La mise en place du marqueur doit suivre l'affichage de la carte
         dans la division (et non pas l'inverse) */
var imageMarqueurQuimper = "beachflag.png";
var marqueurQuimper = new google.maps.Marker({
    position: new google.maps.LatLng(48, -4.1),
    map: maCarte,
    icon: imageMarqueurQuimper
});
```

Le chemin d'accès à l'image qui sert de nouvelle icône de marquage est d'abord défini :

```
var imageMarqueurQuimper = "beachflag.png";
```

Ensuite un paramètre supplémentaire (icon) est intégré dans l'instanciation de marqueur :

```
icon: imageMarqueurQuimper
```

Passons maintenant au cœur du traitement, la mise en place de cercles représentant la population des villes Rennes, Quimper, Vannes et Saint-Brieuc.

La séquence débute par la déclaration d'un tableau mémoire (associatif) dans lequel seront stockés la position géographique des quatre préfectures bretonnes ainsi que leur population :

```
/* Déclaration du tableau des villes */
var listeVilles = {};
```

À titre indicatif, pour la ville de Rennes, les données placées dans le tableau associatif listeVilles sont :

```
/* Paramètres de Rennes */
listeVilles['Rennes'] = {
    position: new google.maps.LatLng(48.0833, -1.6833),
    population: 207178
};
```

Il reste ensuite à balayer le tableau listeVilles pour tracer un cercle de rayon proportionnel à la population pour chacune des villes :

```
/* Balayage du tableau listeVilles pour la construction d'un cercle */
/* de taille proportionnelle à la population de chaque ville */
for (var ville in listeVilles)
{
```

```

/* Construction d'un cercle de rayon calculé */
/* à partir de la population (division par 20) */
var optionsCercle = {
    strokeColor: "#FF0000",
    strokeOpacity: 0.8,
    strokeWeight: 2,
    fillColor: "#FF0000",
    fillOpacity: 0.35,
    map: maCarte,
    center: listeVilles[ville].position,
    radius: listeVilles[ville].population / 20
};
/* Tracé du cercle */
cercleVille = new google.maps.Circle(optionsCercle);
}

```

optionsCercle est un tableau regroupant toutes les options définissant le cercle à tracer pour chaque ville. Les options définies dans notre exemple sont :

- la couleur du pourtour (strokeColor),
- l'opacité du pourtour (strokeOpacity),
- l'épaisseur du trait du pourtour (strokeWeight),
- la couleur de remplissage de l'intérieur du cercle (fillColor),
- l'opacité de la couleur de remplissage de l'intérieur du cercle (fillOpacity),
- la carte sur laquelle placer le cercle (map),
- la position du centre du cercle (center),
- le rayon du cercle (radius).

Enfin le cercle est tracé avec les options définies :

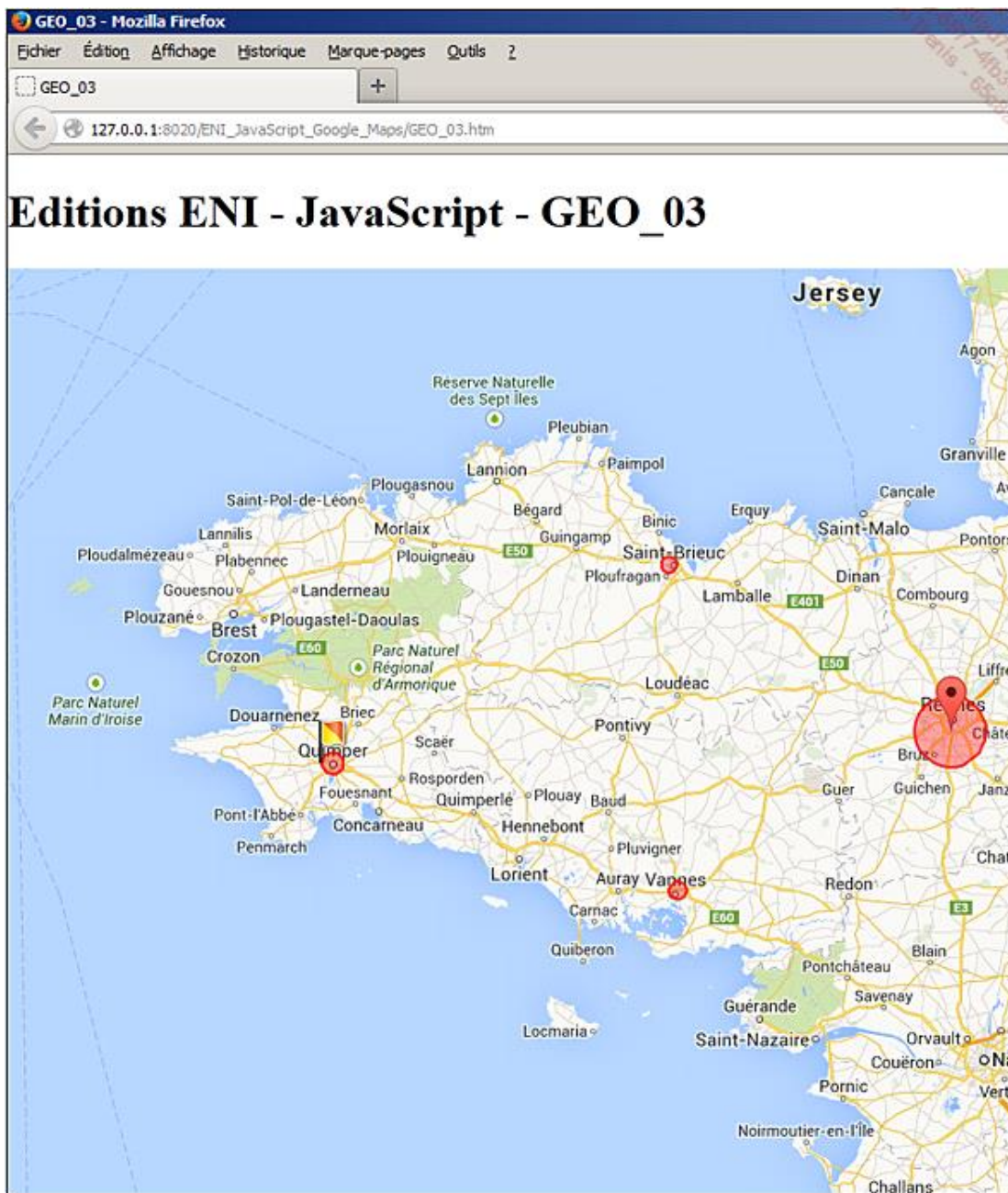
```

/* Tracé du cercle */
cercleVille = new google.maps.Circle(optionsCercle);

```

### **Compte rendu d'exécution**

L'affichage suivant est obtenu à l'exécution du script GEO\_03.htm :



#### 4. Exemple 4 : Affichage de la carte de l'Ouest de la France (informations météorologiques)

Dans ce quatrième exemple, nous allons voir comment rajouter sur une carte une couche informationnelle, des informations météorologiques en temps réel dans notre cas.

##### Section HTML <body>

Aucune nouveauté n'est à signaler par rapport à l'exemple 3, vous y retrouvez donc la division HTML d'affichage de la carte intitulée maCarte.

##### Section HTML <head>

Cette section débute par la définition de quelques styles CSS, utiles pour la présentation des sections HTML html

et body :

```
<!-- Feuille de styles CSS -->
<style>
  /* Style CSS pour les sections html et body */
  html, body
  {
    height: 80%;
    margin: 0;
    padding: 0;
  }
</style>
```

Ensuite nous trouvons l'habituel appel à l'API Google Map :

```
<!-- Appel de l'API Google Map -->
<!-- NB : sensor=false signifie que l'application ne propose pas de
repérage GPS -->
<!--      libraries=weather est la librairie spécifique de la gestion de
la météo -->
<script type="text/JavaScript"
src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=
false&libraries=weather"></script>
```

Dans cet appel, le repérage de la position de l'internaute n'est pas réalisé et par le paramètre `libraries=weather` nous signalons l'utilisation de la librairie Google permettant la récupération des informations météorologiques.

Dans la fonction `initialiserCarte` venant ensuite, après avoir mis en place la carte Google Map dans la division HTML `maCarte` comme suit :

```
/* Mise en place de la carte dans la division maCarte */
var maCarte = new google.maps.Map(document.getElementById("maCarte"),
optionsGoogleMap);
```

voyons comment intégrer la couche informationnelle (layer) évoquée.

Débutons par l'annonce des températures. L'objet utilisé est `WeatherLayer`. Il suffit d'instancier un tel objet et de paramétrer la propriété `temperature Units` comme ceci :

```
/* Définition de la couche (layer) des températures */
var coucheTempératures = new google.maps.weather.WeatherLayer({
  /* Unité de température : CELSIUS ou FARENHEIT */
  temperatureUnits: google.maps.weather.TemperatureUnit.CELSIUS
});
```

`WeatherLayer` est une classe de la librairie `google.maps.weather`. Le choix de l'unité pour les températures a été fixé à `CELSIUS`.

Ensuite cette couche doit être placée sur la carte :

```
/* Placement de la couche des températures sur la carte */  
coucheTemperatures.setMap(maCarte);
```

En ce qui concerne la couche d’affichage des nuages (et du soleil, cela peut aussi arriver en Bretagne), la démarche est sensiblement identique.

L’objet utilisé est, cette fois-ci, `CloudLayer` (sans paramètre) comme suit :

```
/* Définition de la couche (layer) des nuages */  
var coucheNuages = new google.maps.weather.CloudLayer();
```

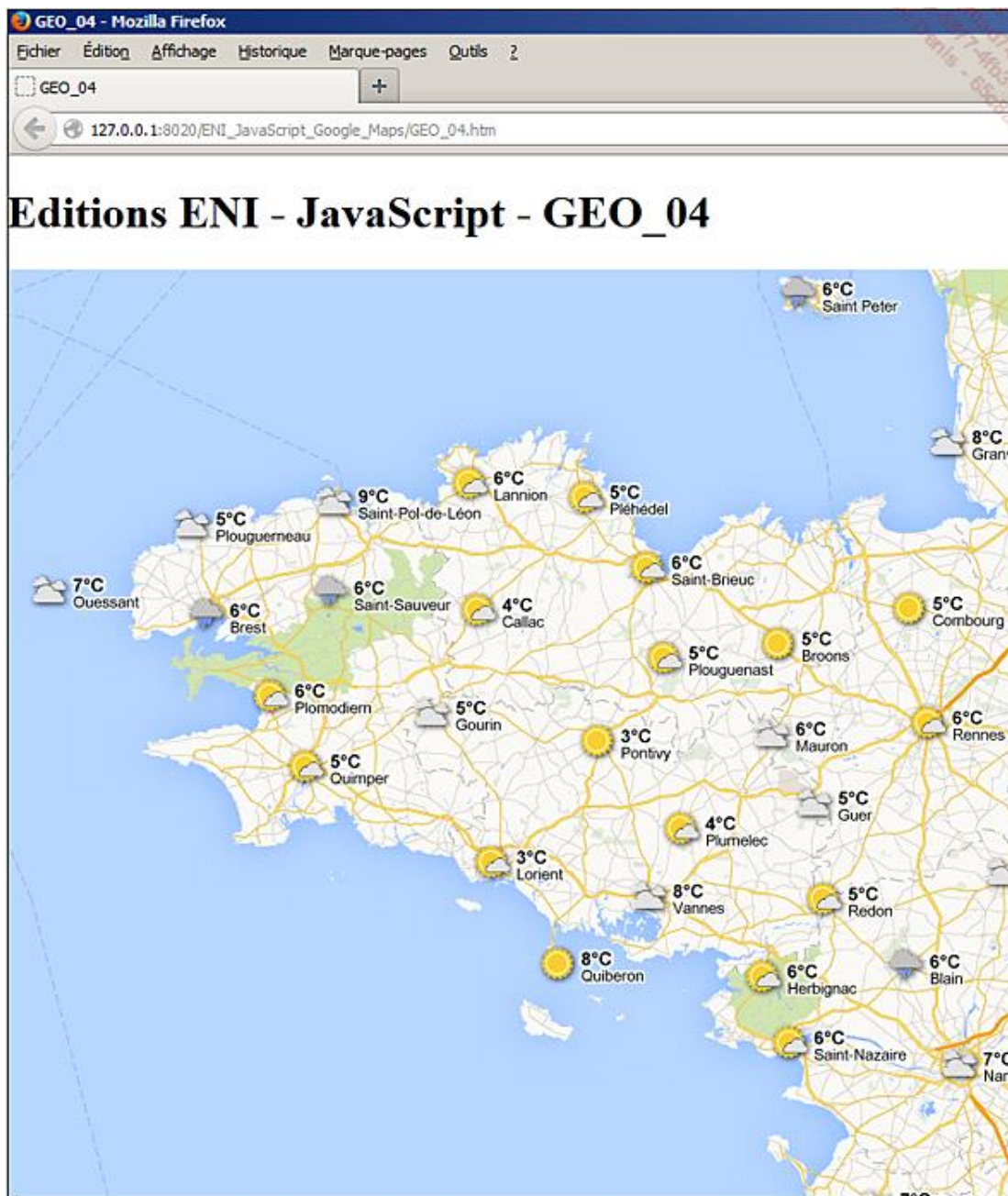
Pour le positionnement des figurines représentant les nuages sur la carte, la méthode `setMap` est encore utilisée :

```
/* Placement de la couche des nuages sur la carte */  
coucheNuages.setMap(maCarte);
```

### **Compte rendu d’exécution**

L’exécution du script `GEO_04.htm` affiche ceci :





## 5. Exemple 5 : Affichage de la carte de Rennes Centre-Sud (couche panorama)

Dans ce nouvel exemple, nous allons étudier une surcouche informationnelle particulièrement intéressante quand nous visitons par exemple une ville avec notre smartphone favori. Il va être possible d'accéder à des informations affichées dans un mini-cadre sur des points d'intérêt.

### Section HTML <body>

Aucune nouveauté n'est à signaler par rapport à l'exemple 4.

### Section HTML <head>

Avant de décrypter notre habituelle fonction `initialiserCarte`, il est nécessaire de voir le contenu d'une feuille de style CSS intégrée :

```

<!-- Feuille de styles CSS -->
<style>
  /* Style CSS pour les sections html et body */
  html, body
  {
    height: 80%;
    margin: 0;
    padding: 0;
  }
  /* Style CSS pour l'affichage des cadres photo */
  #cadrePhoto
  {
    border: 1px solid #ccc;
    width: 300px;
    max-height: 300px;
    background: #fff;
    padding: 5px;
    font-family: Arial;
    font-size: 12px;
  }
</style>

```

Le style #cadrePhoto de cette feuille CSS sert au formatage des cadres dans lesquels les photos des points d'intérêt seront présentées.

L'appel de l'API Google Map est aussi un peu différent par rapport à ce qui a été vu dans les quatre exemples précédents :

```

<!-- Appel de l'API Google Map -->
<!-- NB : sensor=false signifie que l'application ne propose pas de
repérage GPS -->
<!--      libraries=panoramio est la librairie spécifique de la gestion
des lieux à visiter -->
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=
false&libraries=panoramio"></script>

```

Le paramètre libraries=panoramio sert à faire référence à une librairie spécifique pour la gestion des lieux à visiter.

La fonction initialiserCarte commence comme d'habitude par la définition de la position de centrage de la carte (toujours Rennes), la définition des options d'affichage et la mise en place de la carte dans la division HTML nommée maCarte. La seule différence est le facteur de zoom fixé à 16 dans cet exemple.

Comme pour les couches placées sur la carte dans l'exemple 4, plaçons ici une couche nommée PanoramioLayer :

```

/* Définition de la couche (layer) des panoramas */
var couchePanoramas = new google.maps.panoramio.PanoramioLayer();

/* Placement de la couche des panoramas sur la carte */

```

```
couchePanoramas.setMap(maCarte);
```

Ce n'est pas totalement finalisé, il reste à définir le cadre d'affichage des photos et à indiquer la position des photos dans leur cadre;

```
/* Définition du cadre d'affichage des photos */
var cadrePhoto = document.getElementById("cadrePhoto");

/* Position des photos dans les cadres */
map.controls[google.maps.ControlPosition.RIGHT_TOP].push(cadrePhoto);
```

Le cadre des photos est basé sur le style #cadrePhoto décrit dans la feuille de style CSS intégrée. Les photos sont calées dans l'angle droit en haut de ces cadres.

Terminons en associant une action (affichage d'informations complémentaires) à l'événement `click` sur chacune de ces photos :

```
/* Événement clic sur les photos */
google.maps.event.addListener(panoramioLayer, "click", function(photo) {
    /* Définition d'un élément div */
    var div = document.createElement("div");
    /* Définition du lien hypertexte associé à la photo */
    var link = document.createElement("a");
    link.setAttribute("href", photo.featureDetails.url);
    /* Mise en place du lien hypertexte dans la div */
    div.appendChild(link);
});
```

Sans rentrer dans un niveau de détail trop important, disons que dans cette séquence :

- une fonction est décrite à la volée lors d'un clic sur les photos,
- une division HTML nommée `div` est créée,
- un lien hypertexte est associé à la photo.

Le paramètre `photo` de la fonction représente la photo sur laquelle l'opérateur aura cliqué. Cette photo peut avoir été placée sur le site Panoramio (<http://www.panoramio.com>) avec un lien hypertexte.

### **Compte rendu d'exécution**

L'affichage produit par le script `GEO_05.htm` donne ceci :



La section débute par des balises meta que nous connaissons déjà :

```
<!-- Balise meta HTTP-equiv & content -->
<meta HTTP-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- Balise meta définissant la zone affichable -->
<meta name="viewport" content="initial-scale=1.0, user-scalable=yes"/>
<!-- NB : initial-scale=1.0 : Ouverture de la page avec une échelle
à 100 % -->
<!-- user-scalable=yes : Zoom possible de la part de l'utilisateur -->
```

La feuille de style CSS utilisée va être identique à celle vue dans l'exemple 5.

L'appel de l'API Google Map a aussi été vu sous cette forme (exemples 1 à 4) :

```
<!-- Appel de l'API Google Map -->
<!-- NB : sensor=false signifie que l'application ne propose pas de
repérage GPS -->
<script type="text/Javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
```

La fonction `initialiserCarte` débute par le paramétrage du centrage de la carte (sur la ville de Rennes).

Les options de la carte sont des options déjà vues :

- un zoom à 16,
- le centrage,
- le mode d'affichage ROADMAP.

La carte est ensuite affectée à la division HTML habituelle `maCarte`.

La suite est la mise en place du dispositif Street View :

```
/* Mise en place de la carte dans la division maCarte */
var maCarte = new google.maps.Map(document.getElementById("maCarte"),
optionsGoogleMap);

/* Définition des options Street View */
var streetViewOptions =
{
    position: maCarte,
    pov: {heading: 34, pitch: 10}
};

/* Définition de la Street View */
var streetView =
new google.maps.StreetViewPanorama(document.getElementById("maCarte"),
panoramaOptions);
```



```
/* Mise en place de la Street View dans la division maCarte */  
maCarte.setStreetView(streetView);
```

Dans les options de Street View, vous trouvez `position` pour associer le dispositif à la carte et `pov` (pour *point of view* - point de vue).

Le paramètre `pov` est un tableau d'options avec :

- `heading` : angle de rotation de la caméra en degrés (0 = position Nord), dans le sens des aiguilles d'une montre (90 = position Est), la valeur par défaut est 0.
- `pitch` : inclinaison de la caméra (90 degrés = inclinaison vers le haut, -90 degrés = inclinaison vers le bas), la valeur par défaut est 0.

### Compte rendu d'exécution

L'exécution du script `GEO_06.htm` produit cet affichage :

