

Bloque 5

Programación con JavaScript



Anexo: API de Google Maps

Programación con JavaScript
Cefire 2017/2018
Autor: Arturo Bernal Mayordomo

Índice

API de Google Maps.....	3
API estática.....	3
API (dinámica) de Google Maps.....	3
Gestión de eventos en el mapa.....	5
Añadiendo marcadores al mapa.....	6
Midiendo distancias con la librería GMaps Geometry.....	7

API de Google Maps

La [API de Google Maps](#) nos permite integrar un mapa interactivo en una web. Para poder usar gran parte de la API de Google, debemos estar registrados como desarrollador y obtener una clave (key) de desarrollo para nuestra web o app.

API estática

Lo más sencillo, una vez que tenemos la localización, es usar [la API estática de Google](#) y obtener una imagen (estática) centrada de nuestra posición con marcadores (opcional). Vamos a ver un ejemplo y su resultado:

```
navigator.geolocation.getCurrentPosition(function(pos) {  
  var latlon = pos.coords.latitude + "," + pos.coords.longitude;  
  
  var imgUrl = "http://maps.googleapis.com/maps/api/staticmap?center=" +  
    +latlon+"&zoom=14&size=400x300&sensor=false&&markers=color:red%7Clabel:C%7C" +latlon;  
  
  // Ya sólo queda modificar la imagen en nuestro documento donde queramos mostrar el mapa  
  document.getElementById("mapImg").setAttribute('src', imgUrl);  
});
```

Latitude: 38.383342. Longitude: -0.5127781 (accuracy: 30)



API (dinámica) de Google Maps

En primer lugar, debemos crear una clave válida para un proyecto (podemos crear varios proyectos de forma gratuita) en [esta página](#) y que podremos gestionar en cualquier momento desde <https://console.developers.google.com>. Podemos utilizar la clave sin restricciones (desarrollo) o sólo desde páginas alojadas en ciertos dominios o apps para móvil concretas (producción).

Clave de API

Restricciones de clave

Esta clave no tiene restricciones. Para evitar un uso sin autorización y el robo de cuotas, restringela. [Learn more](#)

⚠ Restricciones de aplicación: Ninguna ⚠ Restricciones de API: Ninguna

Restricciones de aplicación

Restricciones de API

Las restricciones de aplicación especifican qué sitios web, direcciones IP o aplicaciones pueden usar esta clave.

Restricciones de aplicación

- ☒ Ninguna
- ☐ URLs de referencia HTTP (sitios web)
- ☐ Direcciones IP (servidores web, tareas cron, etc.)
- ☐ Aplicaciones para Android
- ☐ Aplicaciones para iOS

Antes de cargar el mapa, debemos crear un elemento HTML (en este caso <div id="gmap">) donde se visualizará:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Getting location example</title>
    <link rel="stylesheet" href="style.css">
  </head>

  <body>
    <div>
      <p id="coordinates">Getting location information...</p>
      <div id="gmap"></div>
    </div>
  </body>
</html>
```

El elemento donde se mostrará el mapa debe tener un alto y ancho definidos (tamaño del mapa), o si no, por defecto el alto será 0px.

```
#gmap {
  height: 480px;
  width: 640px;
}
```

Cargar la librería mediante una Promesa

Hay una librería llamada google-maps-promise que carga la librería de Google Maps encapsulando ese proceso (asíncrono) en una promesa. Primero vamos a instalarla:

```
npm i google-maps-promise
```

Y este es el proceso mediante el cual cargaremos un mapa centrándolo en nuestra posición actual (geolocalizada):

```
import { load, urlSettings } from 'google-maps-promise';

urlSettings.key = 'AlzaSyAX2skeHPuTfAp3kDCq8orUcCFHqhPqXJg';
urlSettings.language = 'es';
urlSettings.region = 'ES';
urlSettings.libraries = ['geometry', 'places'];

function loadMap(pos) {
  load()
    .then(GMaps => { // Gmaps (parámetro) es un alias de google.maps (puedes usar ambos)
      document.getElementById("coordinates").textContent = "Latitude" +
        pos.coords.latitude + ". Longitude: " + pos.coords.longitude;

      let gLatLng = new GMaps.LatLng(pos.coords.latitude, pos.coords.longitude);

      let options = {
        zoom: 17, // Map zoom (min: 0, max: 20)
        center: gLatLng, // Centramos el mapa en nuestra posición
        mapTypeId: GMaps.MapTypeId.ROADMAP // Tipo de mapa (SATELLITE, HYBRID, TERRAIN)
      };
      // Esto equivale a llamar a `new google.maps.Map`
      let map = new GMaps.Map(document.getElementById('gmap'), options);
    });
}

window.addEventListener('load', () => {
  navigator.geolocation.getCurrentPosition(loadMap); // Cuando nos geolocaliza, carga el mapa
});
```

Latitude38.3834242. Longitude: -0.5127653



Gestión de eventos en el mapa

Podemos añadir manejadores de eventos al mapa (tenemos que usar los métodos y los [eventos](#) que el API de Google Maps nos facilita). Por ejemplo, eventos como click, dblclick, dragstart, dragend, mousemove, resize, zoom_changed, etc...

Para añadir un manejador de evento usamos **google.maps.event.addListener** que recibe 3 parámetros: el objeto mapa, el nombre del evento (string) y una función (o referencia a una función) que manejará el evento. Vamos a añadir un evento **click** en nuestro mapa, por tanto cuando el usuario haga clic en él, el mapa se centrará donde el usuario haya pulsado.

```
google.maps.event.addListener(map, 'click', event => {
  var coordsP = document.getElementById("coordinates");
  coordsP.textContent = "Me muevo a lat: " + event.latLng.lat() + ", lng: " + event.latLng.lng();
  map.panTo(event.latLng);
});
```

Añadiendo marcadores al mapa

Vamos a ver cómo añadir marcadores por defecto en el mapa. Los marcadores recibirán un conjunto de opciones en formato JSON. Las opciones obligatorias son el objeto del mapa, y el objeto latLng donde se pondrá el marcador. Podemos ver otras opciones [aquí](#). Vamos a añadir un marcador al hacer clic (función **showMap**):

```
// Creamos un marcador rojo en nuestra posición
createMarker(map, gLatLng, 'red');

google.maps.event.addListener(map, 'click', event => {
  var coordsP = document.getElementById("coordinates");
  coordsP.textContent = "Creado un marcador en lat: " + event.latLng.lat() +
    ", lng: " + event.latLng.lng();
  map.panTo(event.latLng);
  // Creamos un marcador verde cada vez que el usuario haga click sobre el mapa
  createMarker(map, event.latLng, 'green');
});
```

Función **createMarker**:

```
// Recibes un objeto de mapa y un objeto latLng, que será donde se cree el marcador
function createMarker(map, latLng, color) {
  var opts = {
    position: latLng,
    map: map,
    icon: 'http://maps.google.com/mapfiles/ms/icons/' + color + '-dot.png'
  };

  var marker = new google.maps.Marker(opts);

  // Añadimos el evento click al marcador también. Este nos mostrará una ventana con la información de las
  // coordenadas
  google.maps.event.addListener(marker, 'click', function(event) {
    var infoWindow = new google.maps.InfoWindow();
    infoWindow.setContent("Marker at lat: " + event.latLng.lat().toFixed(6) +
      ", lng: " + event.latLng.lng().toFixed(6));
    infoWindow.open(map, marker);
  });
}
```

Created a marker at lat: 38.38413056386372, lng: -0.5115294456481934



Midiendo distancias con la librería GMaps Geometry

En el API de Google Maps, hay muchas librerías adicionales que podemos usar para geometría, localización de sitios en un mapa o para dibujar. Hay también otras APIS auxiliares como [Google Maps Directions API](#) que podemos usar para calcular las distancias andando, en bici, en transporte público o en coche.

Vamos a ver cómo se calcula la distancia entre dos puntos en el mapa (sólo las distancias rectas, no las reales si viajamos). Para incluir esta librería en nuestro script sólo tenemos que incluir un parámetro con las librerías extras a cargar en la configuración de google-maps-promise:

```
urlSettings.libraries = ['geometry'];
```

El método `google.maps.geometry.spherical.computeDistanceBetween` (que ya podremos usar) calcula la distancia entre dos objetos `LatLng`:

```
google.maps.event.addListener(map, 'click', function(event) {  
  var coordsP = document.getElementById("coordinates");  
  var dist = google.maps.geometry.spherical.computeDistanceBetween(  
    gLatLng, // Nuestra posición  
    event.latLng // Esta posición  
  );  
  coordsP.innerHTML = "Creado un marcador en: " + event.latLng.lat() +  
    ", lng: " + event.latLng.lng() + "<br>" +  
    "Distancia desde donde estás: " + (Math.round(dist)/1000) + "km";  
  map.panTo(event.latLng);  
  // Crea un marcador verde cada vez que hace click  
  createMarker(map, event.latLng, 'green');  
});
```

Usando Google Places para encontrar lugares

Esta librería te permite encontrar lugares a partir de un punto del mapa o en general. Las búsquedas pueden ser genéricas (restaurante, cine, ...) lo que devolverá

varios resultados, o un lugar o dirección en concreto.

En este ejemplo, vamos a crear un elemento `AutoComplete`, que le sugerirá al usuario lugares o direcciones en base a lo que vaya escribiendo en un input de texto. Este es el código que añadiríamos a la función **loadMap** (recuerda que el código que pongas en otra función debe usar **google.maps**, variable global, en lugar de **Gmaps**).

```
let searchInput = document.getElementById('search'); // Es un <input type="text">

let autocomplete = new Gmaps.places.AutoComplete(searchInput); // Vinculamos el autocompletado
autocomplete.bindTo('bounds', map);

// Creamos un marker y una ventana de información asociada que cambiaremos cada vez que
// seleccionemos un nuevo lugar
let infowindow = new GMaps.InfoWindow();
let marker = new GMaps.Marker({
  map: map
});
GMaps.event.addListener(marker, 'click', function () {
  infowindow.open(map, marker);
});

// Evento que indica que el usuario ha seleccionado un nuevo lugar
google.maps.event.addListener(autocomplete, 'place_changed', () => {
  infowindow.close();

  let place = autocomplete.getPlace();
  if (!place.geometry) return; // Ubicación no válida

  map.setCenter(place.geometry.location);
  map.setZoom(17);

  // Posicionamos el marker en el lugar seleccionado
  marker.setPlace({
    placeId: place.place_id,
    location: place.geometry.location
  });
  marker.setVisible(true);

  // Establecemos el texto de la ventana de información y la abrimos encima del marker
  infowindow.setContent('<div><strong>' + place.name + '</strong><br>' +
    'Latlng: ' + place.geometry.location.lat() + ", " + place.geometry.location.lng() + '<br>' +
    place.formatted_address + '</div>');
  infowindow.open(map, marker);

  coordsP.textContent = "Latitude" + place.geometry.location.lat() +
    ". Longitude: " + place.geometry.location.lng();
});
```


