API Google Maps

APIs de Google

Google API es un **conjunto de APIs** desarrolladas por Google, que permiten la comunicación con los servicios de Google y su integración con otros servicios. Ejemplos de estos incluyen Search, Gmail, Translate o Google Maps. **Aplicaciones de terceros** pueden utilizar estas API para aprovechar o ampliar la funcionalidad de los servicios existentes.

Las API proporcionan funcionalidad como análisis, aprendizaje automático como un servicio (el Prediction API) o el acceso a los datos del usuario (cuando se le da permiso para leer los datos). Otro ejemplo importante es un mapa de Google incrustado en una página web, que se puede conseguir mediante la API de mapas estáticos (**Static maps**), API de sitios (**Places**) o API **Google Earth**.

API Google Maps

Google Maps fue desarrollado originalmente por dos hermanos Daneses, Lars y Jens Rasmussen, co-fundadores de Where 2 Technologies una empresa dedicada a la creación de soluciones de mapeo. La empresa fue adquirida por Google en octubre de 2004, y los dos hermanos luego crearon Google Maps.

Antes de que hubiera una API pública, algunos desarrolladores descubrieron la manera de hackear Google Maps para incorporar los mapas en sus propios sitios web. Esto llevó a Google a la conclusión de que había una necesidad de una API pública, y en junio de 2005 fue lanzada públicamente.

Límites de Uso

Límites de Uso Standard

- Los **usuarios de la API estándar**:

  Libre hasta más de 25.000 cargas de mapa por 24 horas durante 90 días consecutivos

- Habilitar **facturación pay-as-you-go** para desbloquear las cuotas más altas:

  Tras de sobrepasar los límites de uso libre, la facturación de $0.50 USD / 1.000 solicitudes adicionales, hasta un millón por 24 horas

Límites de Uso Premium

- Google Maps API para los clientes de trabajo: Precio basado en el volumen requerido.

¿Cómo funciona Google Maps?

Es sólo **HTML, CSS y JavaScript** trabajando conjuntamente. Los mapas son solo imágenes que se cargan en el fondo a través de peticiones ejecutadas por la tecnología de **AJAX**, y se insertan en un en la página HTML. Mientras navegas en el mapa, el API envía información acerca de las nuevas coordenadas y los niveles de "zoom" de el mapa a través de AJAX y esto retorna las imágenes.

El API consiste de archivos **JavaScript** que contienen las **clases**, **métodos** y **propiedades** que se usan para el comportamiento de los mapas.

¿Cómo se usan?

Veremos a continuación como trabajar con la última versión creada al momento, la versión 3.

Las **coordenadas** están expresadas usando **números decimales separados por coma**. La latitud siempre precede la longitud. La latitud es positiva si es N y negativa si es S. La longitud es positiva si es E y negativa si es O.

En los mapas físicos, las coordenadas están expresadas en grados, así que la posición de Benidorm sería:

38° 32′ 03″ N, 0° 07′ 53″ O

La forma de convertir estos datos a decimales sería:

(38° 32′ 03″ N) = (38 + (32 / 60) + (03 / 3600)) = 38,534167
(0° 07′ 53″ O) = -(0 + (07 / 60) + (53 / 3600)) = -0,131389

La longitud se multiplica por negativo, porque está a la izquierda (oeste) del punto 0,0.

¿Cuál es el máximo número de decimales?

Google maps **no se limita** a cierta cantidad de decimales. Sin embargo, según unas pruebas hechas, se notó que utilizar números **mayores a 6 decimales es una pérdida de tiempo**. Así pues, Google estableció en varios métodos que la mayor cantidad a trabajar es 6 decimales. Es decir, cuando vayamos a establecer los decimales se puede hacer así:

- **5 a 6 decimales**: es el máximo que debemos usar para ser específicos.
- **4 decimales**: para algún detalle en el mapa.
- **3 decimales**: es bueno para centrar ciudades.
- **2 decimales**: es apropiado para centrar países o estados, tal vez 3 para países pequeños.

Preparando el área de trabajo

Utilizaremos el 100% de ancho y el 100% de alto del navegador. También nos limitaremos a explicar solo la parte del API. La referencia del último API que contiene las clases, métodos y propiedades puede encontrarse en esta dirección:

https://developers.google.com/maps/documentation/javascript/reference

Código de los ejemplos:

Fichero **HTML**:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8" />
<link rel="stylesheet" href="googleMaps.css">
<title>Prova Google Maps API</title>
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?
sensor=false&amp;language=es"></script>
<script type="text/javascript" src="js/map.js"></script>
</head>
<body>
<div id="map"></div>
</body>
</html>
```

Fichero **CSS**:

```
*{
	margin: 0;
	padding: 0;
}

html, body, #map{
	width: 100%;
	height: 100%;
}
```

Fichero **JS**:

```
function mapaGoogle(){
        var options = {
                zoom: 8
                , center: new google.maps.LatLng(38.53, -0.13)
                , mapTypeId: google.maps.MapTypeId.SATELLITE
        };
        var map = new google.maps.Map(document.getElementById('map'), options);
};

window.addEventListener('load',mapaGoogle,false)
```

En el siguiente código HTML, que carga los scripts de javascript:

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false&;amp;language=es"></script>
```

La dirección apunta al API, pero también se requiere pasar una cadena de consulta con la clave **sensor**. Esto le indica a Google si el dispositivo que usa el mapa, tiene un **dispositivo que determina la geolocalización**, como por ejemplo un GPS. Es **obligatorio** mencionarlo indicando como valor si es false o true. Falso para los que no lo usan y cierto para los que sí lo usan. Esto lo usa Google para proveer estadísticas a sus proveedores. No tiene que ver con habilitarlo para la geolocalización.

También se añadió el lenguaje, aunque el API trata de determinar cual es el lenguaje a mostrar, pero se puede especificar usando la clave `language`.

Es importante que este código:

```
<script type="text/javascript" src="map.js"></script>
```

se encuentre debajo del elemento `<script>` que incluye el API, para que así se puedan cargar todas las clases, métodos y propiedades a usar. Las clases, métodos y propiedades comienzan con `google.maps`. A esto se le conoce como `namespace`.

¿Qué es lo que hace?

El código de map.js nos trae el mapa de Benidorm. Indicamos que queremos que el mapa se muestre una vez se haya cargado toda la información con la escucha:

```
window.addEventListener('load',mapaGoogle,false);
```

Para iniciar el mapa hacemos uso de la clase `google.maps.Map()`. A esta se le introducen dos argumentos:

- La **referencia hacia el elemento que mostrará el mapa**, en este caso el elemento `<div>` con el atributo id que tiene como valor `map`.

```
var elementMapa=document.getElementById('map');
```

- Una notación literal llamada **MapOptions** que contiene la **configuración inicial** para mostrar el mapa como por ejemplo:
    - El “zoom”.
    - Dónde está el centro.
    - Qué tipo de mapa deseamos mostrar.

MapOptions

`MapOptions` contiene la información de **cómo queremos ver el mapa** y **cómo queremos que se comporte**. Hay tres **propiedades** que son **obligatorias**:

- **zoom**: Define el **“zoom” inicial**. Debe ser un número **entre el 1 y el 23**. El 1 es el mapa totalmente alejado y 23 es lo más cercano posible.
- **center**: Define el **centro** del **mapa** con las **coordenadas**. Las coordenadas deben indicarse usando el método google.maps.LatLng(latitud, longitud).
- **mapTypeId**: Define que tipo de mapa se desea mostrar al inicio.

Pero hay muchas otras propiedades opcionales para `MapOptions`:

- **zoomControl**: muestra botones "+" i "-" para cambiar el **nivel de zoom** del mapa. Aparece por defecto en la esquina inferior derecha del mapa.

  Valores: `true`|`false`, por defecto `false`.

- **RotateControl**: proporciona una combinación de **inclinación y opciones de rotación de mapas** que contienen imágenes oblicuas. Pero solo estará disponible si en la localización existen imágenes en 45º. Este control aparece de forma predeterminada en la parte inferior derecha del mapa.

  Valores: `true`|`false`, por defecto `true`.

- **keyboardShortcuts**: Habilita o inhabilita el **uso del teclado**. Las teclas a usar son las flechas para mover el mapa y +/- para el “zoom”.

  Valores: true|false, por defecto `true`.

- **disableDoubleClickZoom**: Habilita o inhabilita el **doble click** del ratón para hacer “**zoom**”.

  Valores: `true`|`false` por defecto `false`.

- **draggable**: Habilita o inhabilita el poder **arrastrar el mapa**.

Valores: `true`|`false`, por defecto `true`.

- **scrollwheel**: Habilita o inhabilita el poder **hacer "zoom" con la rueda del ratón**.

  Valores: `true`|`false`, por defecto `true`.

- **draggableCursor**: Indica que **tipo de cursor** deseas mostrar cuando el ratón está **encima del mapa**. El valor es del tipo cadena y pueden ser los que una computadora tiene por defecto y la mayoría de los que están mencionado en el siguiente listado o puede ser uno personalizado y la ruta puede ser local (en el servidor de la aplicación) o una dirección web externa.

- **auto**
- **crosshair**
- **default**
- **pointer**
- **move**
- **e-resize**
- **ne-resize**
- **nw-resize**
- **n-resize**
- **se-resize**
- **sw-resize**
- **s-resize**
- **w-resize**
- **text**
- **wait**
- **help**
- **progress**

- **draggingCursor**: Indica que **tipo de cursor** deseas mostrar cuando el ratón está **presionado en el mapa**. El valor es del tipo cadena y pueden ser los mismos tipos que en el atributo anterior.

- **backgroundColor**: Esta propiedad afecta el **color del fondo** del contenedor. Típicamente se ve cuando se arrastra el mapa o cuando carga al inicio. Puedes usar un valor hexadecimal o la forma estándar (red, yellow, green, blue, etc). Por defecto el color es con el valor hexádecimal #E5E3DF

- **noClear**: Habilita o inhabilita que se **sobreescriba lo que haya en el contenedor**. Normalmente la forma como se trabaja para colocar contenido encima del contenedor es usando un elemento fuera del contenedor que muestra el mapa y con CSS se coloca en el lugar deseado en el mapa.

  Valores: `true`|`false`, por defecto `false`.

- **disableDefaultUI**: Habilita o inhabilita **mostrar el UI que viene predefinido**.

  Valores: `true`|`false`, por defecto `false`.

- **mapTypeControl**: Habilita o inhabilita el control de **tipo de mapa**.

  Valores: `true`|`false`, por defecto `true`.

- **mapTypeControlOptions**: Son las **opciones** de visualización iniciales del **control del tipo de mapa.**

  Valores de tipo: (`ROADMAP` | `SATELLITE` | `HYBRID` | `TERRAIN`)

- **navigationControl**: Habilita o inhabilita el **control de navegación**.

  Valores: `true`|`false`, por defecto `true`.

- **navigationControlOptions**: Son las **opciones** de visualización iniciales del **control de navegación**.

- **scaleControl**: Habilita o inhabilita el **control de escala**.

  Valores: `true`|`false`, por defecto `true`.

- **scaleControlOptions**: Son las **opciones** de visualización iniciales del **control de escala**.

- **streetViewControl**: Habilita o inhabilita el **hombrecito de "Street View"**. Está disponible **en ciertas áreas**.

  Valores: `true`|`false`, por defecto `false`.

Modificamos el código de las opciones en el archivo map.js

```
var options = {
		zoom: 8
		, center: new google.maps.LatLng(38.53, -0.13)
		, mapTypeId: google.maps.MapTypeId.SATELLITE

		, backgroundColor: '#ffffff'
		, noClear: true
		, disableDefaultUI: true
		, keyboardShortcuts: false
		, disableDoubleClickZoom: true
		, draggable: false
		, scrollwheel: false
		, draggableCursor: 'move'
		, draggingCursor: 'move'

		, mapTypeControl: true
		, mapTypeControlOptions: {
			style: google.maps.MapTypeControlStyle.HORIZONTAL_MENU
			, position: google.maps.ControlPosition.TOP_LEFT
			, mapTypeIds: [
				google.maps.MapTypeId.SATELLITE
			]
		}
		, navigationControl: true
		, streetViewControl: true
		, navigationControlOptions: {
			position: google.maps.ControlPosition.TOP_RIGHT
			, style: google.maps.NavigationControlStyle.ANDROID
		}
		, scaleControl: true
		, scaleControlOptions: {
			position: google.maps.ControlPosition.TOP_LEFT
```

```
                , style: google.maps.ScaleControlStyle.DEFAULT
        }
};
```

Modificar los valores ya asignados

Hasta ahora, hemos iniciado los valores **directamente en el MapOptions** para iniciar el mapa. Después de que se hayan cargado, podemos **modificarlos** con el **método `setOptions`**.

Podemos modificar la mayoría de las propiedades. Solo **tres no son modificables**: `noClear`, `backgroundColor` y `disableDefaultUI`, por lo que debemos estar seguros qué debemos hacer con ellos al inicio. El `setOptions` es de gran utilidad para **interactuar con el usuario**.

Veamos un ejemplo, que añadimos a nuestro fichero:

```
map.setOptions({
        zoom: 10
        , center: new google.maps.LatLng(38.53, -0.13)
        , mapTypeId: google.maps.MapTypeId.TERRAIN
        , keyboardShortcuts: true
        , disableDoubleClickZoom: false
        , draggable: true
        , scrollwheel: true
        , draggableCursor: 'hand'
        , draggingCursor: 'hand'
        , mapTypeControlOptions: {
                style: google.maps.MapTypeControlStyle.DROPDOWN_MENU
                , position: google.maps.ControlPosition.TOP_RIGHT
                , mapTypeIds: [
                        google.maps.MapTypeId.ROADMAP
                        , google.maps.MapTypeId.SATELLITE
                ]
        }
        , navigationControlOptions: {
                position: google.maps.ControlPosition.TOP_LEFT
                , style: google.maps.NavigationControlStyle.ZOOM_PAN
        }
        , scaleControlOptions: {
                position: google.maps.ControlPosition.BOTTOM_LEFT
                , style: google.maps.ScaleControlStyle.DEFAULT
        }
});
```

Getters y Setters

Existen unos métodos que nos permiten solo **modificar** y/o **obtener** los resultados de las **propiedades** que son **obligatorias**.

- `getZoom()`
- `setZoom(1-23)`
- `getCenter()`
- `setCenter(google.maps.LatLng(latitud, longitud))`
- `getMapTypeId()`
- `setMaptTypeId(google.maps.MapTypeId.*)`

Por Ejemplo:

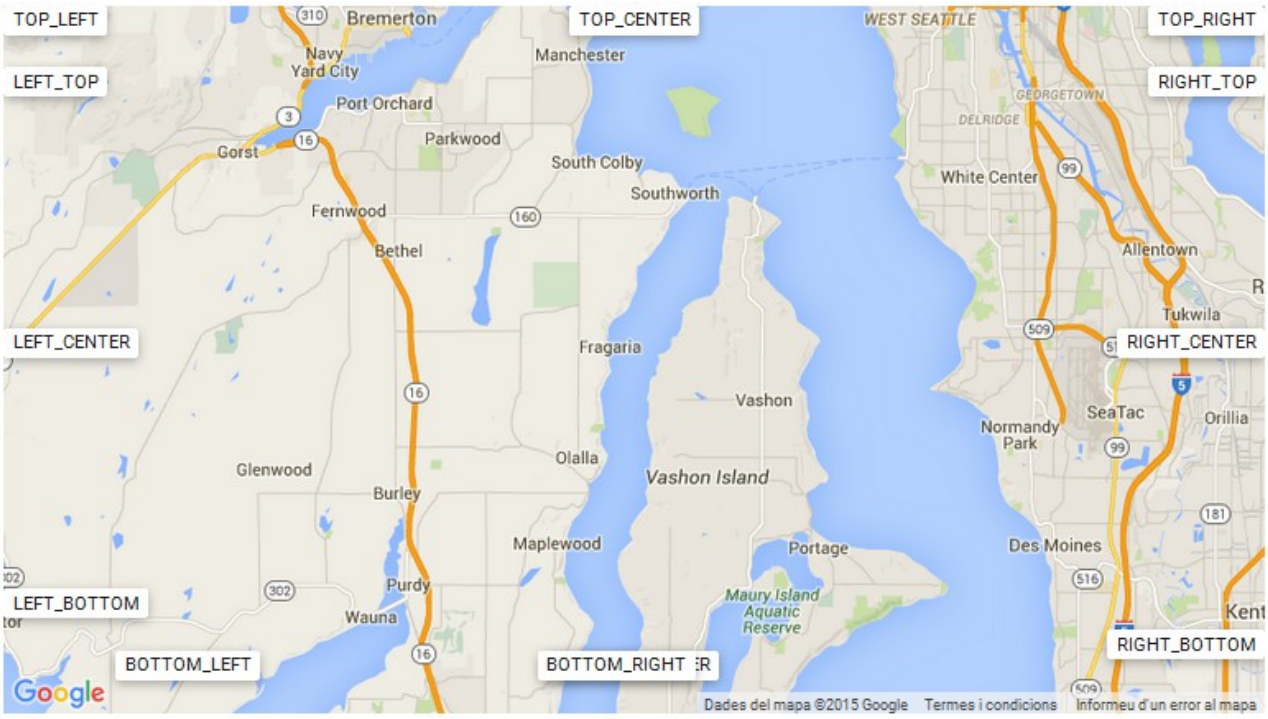
```
map.setZoom(9);
var zoomLevel = map.getZoom();

map.setCenter(new google.maps.LatLng(18.17, -66.3));
var centerOfMap = map.getCenter();

map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
var mapTypeIdOfMap = map.getMapTypeId();
```

Posicionando los elementos

En las opciones de los **controles** podemos **ubicarlos** de las siguientes maneras:

- **TOP_CENTER** indica que el control debe ser colocado a lo largo del centro de la parte superior del mapa.
- **TOP_LEFT** indica que el control debe ser colocado en la parte superior izquierda del mapa, con cualquier subelemento del control de "flow" hacia el centro de la parte superior.
- **TOP_RIGHT** indica que el control debe ser colocado en la parte superior derecha del mapa, con cualquier subelemento del control de "flow" hacia el centro de la parte superior.
- **LEFT_TOP** indica que el control debe ser colocado en la parte superior izquierda del mapa, pero debajo de los elementos `TOP_LEFT`.
- **RIGHT_TOP** indica que el control debe ser colocado en la parte superior derecha del mapa, pero por debajo de cualquier elemento `TOP_RIGHT`.
- **LEFT_CENTER** indica que el control debe ser colocado a lo largo del lado izquierdo del mapa, centrado entre las posiciones `TOP_LEFT` y `BOTTOM_LEFT`.

- **RIGHT_CENTER** indica que el control debe ser colocado en la parte derecha del mapa, centrado entre el `TOP_RIGHT` y la posición `BOTTOM_RIGHT`.
- **LEFT_BOTTOM** indica que el control debe ser colocado en la parte inferior izquierda del mapa, pero por encima de cualquier elemento `BOTTOM_LEFT`.
- **RIGHT_BOTTOM** indica que el control debe ser colocado en la parte inferior derecha del mapa, pero por encima de cualquier elemento `BOTTOM_RIGHT`.
- **BOTTOM_CENTER** indica que el control debe ser colocado a lo largo del centro de la parte inferior del mapa.
- **BOTTOM_LEFT** indica que el control debe ser colocado en la parte inferior izquierda del mapa, con cualquier subelemento del control de "flow" hacia la parte inferior central.
- **BOTTOM_RIGHT** indica que el control debe ser colocado en la parte inferior derecha del mapa, con cualquier subelemento del control de "flow" hacia la parte inferior central.



Añadir un marcador

Un **marcador** identifica una **ubicación en un mapa**. Por defecto, un marcador utiliza una imagen estándar. Los marcadores pueden mostrar imágenes **personalizadas**, conocidos generalmente como "iconos". Los marcadores y los iconos son objetos de tipo `Marker`. Se puede configurar un icono personalizado dentro del constructor, o llamando `setIcon()` en el marcador.

El constructor `google.maps.Marker` toma como parámetro un solo objeto de opciones `Marker`, que especifica las propiedades iniciales del marcador.

Los siguientes campos son particularmente **importantes** y son comúnmente establecidos en la construcción de un marcador:

- **position** (**obligatorio**) especifica un **LatLng** que indica la localización inicial del marcador.
- **map** (**opcional**) especifica el **mapa en el que colocar el marcador**. Si no se especifica el mapa en la construcción del marcador, se crea el marcador pero no se une a (o aparece en) el mapa. Se puede añadir el marcador más tarde llamando al método `setMap()`.

  Ejemplo:

```
var marker = new google.maps.Marker({
   position: myLatLng,
   map: map,
   title: 'Benidorm'
 });
```

Se añade al mapa definido como `map` de nuestro ejemplo. También se puede añadir después de la definición del marcador, el resultado es el mismo

```
var marker = new google.maps.Marker({
   position: myLatlng,
   title:"Benidorm"
});

marker.setMap(map);
```