**API DE GOOGLE MAPS V.3**

Contenido

[Cómo obtener una clave de API 1](#_Toc389472713)

[Creación de un simple mapa (ver uso de Api key) 2](#_Toc389472714)

[COMO CARGAR EL API EN FORMA ASINCRONA 3](#_Toc389472715)

[TIPOS DE MAPAS 4](#_Toc389472716)

[El objeto "Map" 5](#_Toc389472717)

[EVENTOS 5](#_Toc389472718)

[EVENTOS DE LA INTERFAZ DE USUARIO 6](#_Toc389472719)

[CAMBIOS DE ESTADO 6](#_Toc389472720)

[EJEMPLO DE EVENTOS Y CAMBIOS DE ESTADO 6](#_Toc389472721)

[Descripción general de los controles del mapa 7](#_Toc389472722)

[Cómo añadir controles al mapa 8](#_Toc389472723)

[Un ejemplo de un control personalizado 9](#_Toc389472724)

[Marcadores 10](#_Toc389472725)

[POLIGONOS 12](#_Toc389472726)

[INFO WINDOWS (INFORMACION DE MARCADORES) 13](#_Toc389472727)

[CALCULO DE RUTA A TRAVES DE DOS PUNTOS 15](#_Toc389472728)

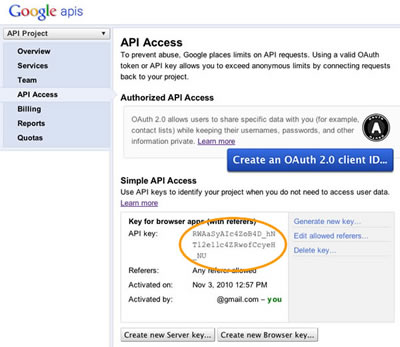
[CALCULO DE RUTA CON INDICACIONES AL COSTADO 17](#_Toc389472729)

[EJEMPLOS EN LA PÁGINA OFICIAL 19](#_Toc389472730)

## Cómo obtener una clave de API

Para crear tu clave de API:

1. Accede a la página de la consola de las API (<https://code.google.com/apis/console>) e inicia sesión con tu cuenta de Google.
2. Haz clic en el enlace de **servicios** en el menú de la izquierda.
3. Activa el servicio de la **versión 3 del API de Google Maps**.
4. Haz clic en el enlace de **acceso al API** en el menú de la izquierda. Tu clave de API está disponible desde la página de **acceso al API**, en la sección de **acceso al API sencilla**. Las aplicaciones del API de Google Maps utilizan la **clave para aplicaciones del navegador**.



Antes era necesario usar esta Key en la siguiente línea:

src="http://maps.googleapis.com/maps/api/js?key=*YOUR\_API\_KEY*&sensor=*SET\_TO\_TRUE\_OR\_FALSE*">

Pero ahora ya no es necesario como se ve en el código de ejemplo, igual se puede seguir utilizando la Key si es que la tienes.

## Creación de un simple mapa (ver uso de Api key)

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Simple Map</title>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
    </style>  
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>  
    <script>  
var map;  
function initialize() {  
  var mapOptions = {  
    zoom: 8,  
    center: new google.maps.LatLng(-34.397, 150.644)  
  };  
  map = new google.maps.Map(document.getElementById('map-canvas'),  
      mapOptions);  
}  
  
google.maps.event.addDomListener(window, 'load', initialize);  
  
    </script>  
  </head>  
  <body>  
    <div id="map-canvas"></div>  
  </body>  
</html>

**CONSIDERACIONES:** Ten en cuenta que algunas de las CSS que funcionan en el modo chapucero no son válidas en el modo estándar. En concreto, todos los tamaños basados en porcentajes deben heredarse de los elementos de bloque principales, y si cualquiera de estos antecesores no puede especificar un tamaño, se supondrá un tamaño de 0 x 0 píxeles. Por esta razón, incluimos la declaración <style>:

### COMO CARGAR EL API EN FORMA ASINCRONA

Esto hace que el api de Java Script de Google Maps se cargue después de toda la página. Es una recomendación que nos hace Google.

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Asynchronous Loading</title>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
    </style>  
    <script>  
function initialize() {  
  var mapOptions = {  
    zoom: 8,  
    center: new google.maps.LatLng(-34.397, 150.644)  
  };  
  
  var map = new google.maps.Map(document.getElementById('map-canvas'),  
      mapOptions);  
}  
  
function loadScript() {  
  var script = document.createElement('script');  
  script.type = 'text/javascript';  
  script.src = 'https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false&' +  
      'callback=initialize';  
  document.body.appendChild(script);  
}  
  
window.onload = loadScript;  
  
    </script>  
  </head>  
  <body>  
    <div id="map-canvas"></div>  
  </body>  
</html>

### 

### TIPOS DE MAPAS

También debes establecer expresamente un tipo de mapa inicial en este momento.

mapTypeId: google.maps.MapTypeId.ROADMAP

Se admiten los siguientes tipos de mapas:

* ROADMAP, que muestra los mosaicos normales en 2D predeterminados de Google Maps.
* SATELLITE muestra imágenes de satélite.
* HYBRID muestra una mezcla de mosaicos fotográficos y una capa de mosaicos para los elementos del mapa más destacados (carreteras, nombres de ciudades, etc.).
* TERRAIN muestra mosaicos de relieve físico para indicar las elevaciones del terreno y las fuentes de agua (montañas, ríos, etc.).

## El objeto "Map"

var map = new google.maps.Map( document.getElementById("map\_canvas"),    mapOptions);

La clase de JavaScript que representa a los mapas es Map. Cada objeto de esta clase define un único mapa en una página. (Puedes crear más de una instancia de esta clase; cada objeto definirá un mapa independiente en la página). Creamos una nueva instancia de esta clase mediante el operador new de JavaScript.

Al crear una nueva instancia de mapa, se especifica un elemento HTML <div> en la página como contenedor para el mapa. Los nodos HTML son elementos secundarios del objeto document de JavaScript. Se obtiene una referencia a este elemento mediante el método document.getElementById().

Este código permite definir una variable (denominada map) y asignar dicha variable a un nuevo objeto Map, además de transmitir opciones definidas en el objeto mapOptionsliteral. Estas opciones se utilizarán para inicializar las propiedades del mapa. A continuación se muestra la definición de la función Map(), conocida como *constructor*:

|  |  |
| --- | --- |
| **Constructor** | **Descripción** |
| Map(mapDiv:Node, opts?:*[MapOptions](https://developers.google.com/maps/documentation/javascript/tutorial?hl=es" \l "MapOptions)*) | Crea un mapa nuevo dentro del contenedor HTML en cuestión, que suele ser un elemento DIV, mediante los parámetros (opcional) que se especifiquen. |

### EVENTOS

JavaScript en el navegador está orientado a eventos, lo que significa que JavaScript responde a las interacciones generando eventos y espera que un programa detecte los eventos interesantes. El modelo de evento de la versión 3 del API de Google Maps es similar al utilizado en la versión 2 del API, aunque se han modificado muchos aspectos de su funcionamiento. Hay dos tipos de eventos:

* Los eventos de usuario (como, por ejemplo, los eventos de ratón "click") se propagan desde el DOM hasta el API de Google Maps. Estos eventos son distintos e independientes de los eventos DOM estándar.
* Las notificaciones de cambio de estado de MVC reflejan los cambios en los objetos del API de Google Maps y se denominan mediante una convenciónproperty\_changed.

Cada objeto del API de Google Maps exporta una determinada cantidad de eventos con nombres. Los programas interesados en determinados eventos registran **detectores de eventos** de JavaScript para estos eventos y ejecutan código al recibirlos mediante el registro de controladores de eventos addListener() en el espacio de nombregoogle.maps.event. Los desarrolladores de la versión 2 del API de Google Maps están familiarizados con este uso.

Para obtener una lista completa de eventos, consulta la [Referencia del API de Google Maps](https://developers.google.com/maps/documentation/javascript/reference?hl=es). Los eventos aparecen en una sección independiente para cada objeto que contiene eventos.

### EVENTOS DE LA INTERFAZ DE USUARIO

Algunos objetos del API de Google Maps están diseñados para responder a los eventos de usuario como los de ratón o los de teclado. Por ejemplo, un objetogoogle.maps.Marker puede detectar los siguientes eventos de usuario:

* 'click'
* 'dblclick'
* 'mouseup'
* 'mousedown'
* 'mouseover'
* 'mouseout'

Estos eventos pueden tener el aspecto de eventos DOM estándar pero, realmente, forman parte del API de Google Maps. Dado que los distintos navegadores implementan distintos modelos de eventos DOM, el API de Google Maps ofrece estos mecanismos que permiten detectar eventos DOM y responder a ellos sin necesidad de gestionar las diversas peculiaridades de cada navegador. Estos eventos suelen incluir argumentos en el evento que destacan algún estado de la interfaz de usuario (como, por ejemplo, la posición del ratón).

### CAMBIOS DE ESTADO

Los objetos MVC suelen contener estado. Cuando cambia la propiedad de un objeto, el API activa un evento que la propiedad ha cambiado. Por ejemplo, el API activa un eventozoom\_changed en un mapa cuando cambia el nivel de zoom del mismo. Para interceptar estos cambios de estado, registra también los controladores de eventosaddListener() en el método de espacio de nombre event.

Los eventos de usuario y los cambios de estado de MVC pueden parecer similares pero, generalmente, se recomienda tratarlos de forma diferente en el código. Los eventos MVC, por ejemplo, no incluyen argumentos en sus eventos. Se recomienda inspeccionar la propiedad que se ha modificado en un cambio de estado de MVC mediante la ejecución del método getProperty adecuado de ese objeto.

### EJEMPLO DE EVENTOS Y CAMBIOS DE ESTADO

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Simple click event</title>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
    </style>  
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>  
    <script>  
function initialize() {  
  var mapOptions = {  
    zoom: 4,  
    center: new google.maps.LatLng(-25.363882, 131.044922)  
  };  
  
  var map = new google.maps.Map(document.getElementById('map-canvas'),  
      mapOptions);  
  
  var marker = new google.maps.Marker({  
    position: map.getCenter(),  
    map: map,  
    title: 'Click to zoom'  
  });  
  
  google.maps.event.addListener(map, 'center\_changed', function() {  
    // 3 seconds after the center of the map has changed, pan back to the  
    // marker.  
    window.setTimeout(function() {  
      map.panTo(marker.getPosition());  
    }, 3000);  
  });  
  
  google.maps.event.addListener(marker, 'click', function() {  
    map.setZoom(8);  
    map.setCenter(marker.getPosition());  
  });  
}  
  
google.maps.event.addDomListener(window, 'load', initialize);  
  
    </script>  
  </head>  
  <body>  
    <div id="map-canvas"></div>  
  </body>  
</html>

## Descripción general de los controles del mapa

Los mapas que se muestran mediante el API de Google Maps contienen elementos de interfaz de usuario que permiten interactuar con el mapa. Estos elementos se denominan*controles* y puedes incluir variaciones de ellos en la aplicación que hayas creado con el API de Google Maps. También puedes optar por no hacer nada y dejar que el API de Google Maps se ocupe del comportamiento de todos los controles.

El API de Google Maps dispone de varios controles integrados que puedes emplear en tus mapas:

* El ***control de zoom*** muestra un control deslizante (para mapas grandes) o pequeños botones "+/-" (para mapas pequeños) que permiten controlar el nivel de zoom del mapa. Este control aparece en la esquina superior izquierda del mapa de forma predeterminada en dispositivos no táctiles o en la esquina inferior izquierda del mapa en los dispositivos táctiles.
* El ***control de desplazamiento*** muestra botones para desplazarse por el mapa. Este control aparece en la esquina superior izquierda del mapa de forma predeterminada en dispositivos no táctiles. El control de desplazamiento también te permite girar imágenes de 45º, si están disponibles.
* El ***control de escala*** muestra un elemento de escala del mapa. Este control está inhabilitado de forma predeterminada.
* El ***control MapType*** permite al usuario alternar entre los diferentes tipos de mapas, como ROADMAP (mapa de carreteras) y SATELLITE (satélite). Este control aparece en la esquina superior derecha del mapa de forma predeterminada.
* El ***control de Street View*** incluye un icono de hombrecito naranja que se puede arrastrar hasta el mapa para habilitar la función de Street View. Este control aparece en la esquina superior izquierda del mapa de forma predeterminada.
* El ***control de giro*** incluye un icono circular pequeño que te permite girar mapas que contienen imágenes oblicuas. Este control aparece en la esquina superior izquierda del mapa de forma predeterminada (para obtener más información, consulta la documentación sobre [imágenes de 45º](https://developers.google.com/maps/documentation/javascript/maptypes?hl=es#45DegreeImagery)).
* El ***control de mapa general*** muestra un mapa general en miniatura que refleja la ventana gráfica del mapa actual dentro de una zona más amplia. Este control aparece en la esquina inferior derecha del mapa y se muestra contraído de forma predeterminada.

### Cómo añadir controles al mapa

Si quieres, puedes confeccionar tu interfaz mediante la eliminación, la adición y la modificación del comportamiento y de los controles de la interfaz de usuario y asegurarte de que las futuras actualizaciones no modifiquen este comportamiento. Si solo quieres añadir o modificar comportamientos existentes, deberás asegurarte de que el control se haya añadido explícitamente a tu aplicación.

Algunos controles aparecen en el mapa de forma predeterminada, mientras que otros no se mostrarán a menos que lo solicites explícitamente. La adición o la eliminación de controles del mapa se especifica mediante los siguientes campos del objeto Map options; campos que deberás establecer en true para mostrar los controles o en false para ocultarlos:

{  
  panControl: boolean,  
  zoomControl: boolean,  
  mapTypeControl: boolean,  
  scaleControl: boolean,  
  streetViewControl: boolean,  
  overviewMapControl: boolean  
}

### Un ejemplo de un control personalizado

El control que se muestra a continuación es sencillo (aunque no especialmente útil) y combina los patrones indicados anteriormente. Este control responde a los eventos 'click'DOM centrando el mapa en una ubicación predeterminada:

var map;  
var chicago = new google.maps.LatLng(41.850033, -87.6500523);  
  
/\*\*  
 \* The HomeControl adds a control to the map that simply  
 \* returns the user to Chicago. This constructor takes  
 \* the control DIV as an argument.  
 \*/  
  
function HomeControl(controlDiv, map) {  
  
  // Set CSS styles for the DIV containing the control  
  // Setting padding to 5 px will offset the control  
  // from the edge of the map.  
  controlDiv.style.padding = '5px';  
  
  // Set CSS for the control border.  
  var controlUI = document.createElement('div');  
  controlUI.style.backgroundColor = 'white';  
  controlUI.style.borderStyle = 'solid';  
  controlUI.style.borderWidth = '2px';  
  controlUI.style.cursor = 'pointer';  
  controlUI.style.textAlign = 'center';  
  controlUI.title = 'Click to set the map to Home';  
  controlDiv.appendChild(controlUI);  
  
  // Set CSS for the control interior.  
  var controlText = document.createElement('div');  
  controlText.style.fontFamily = 'Arial,sans-serif';  
  controlText.style.fontSize = '12px';  
  controlText.style.paddingLeft = '4px';  
  controlText.style.paddingRight = '4px';  
  controlText.innerHTML = '<strong>Home</strong>';  
  controlUI.appendChild(controlText);  
  
  // Setup the click event listeners: simply set the map to Chicago.  
  google.maps.event.addDomListener(controlUI, 'click', function() {  
    map.setCenter(chicago)  
  });  
}  
  
function initialize() {  
  var mapDiv = document.getElementById('map\_canvas');  
  var mapOptions = {  
    zoom: 12,  
    center: chicago,  
    mapTypeId: google.maps.MapTypeId.ROADMAP  
  }  
  map = new google.maps.Map(mapDiv, mapOptions);  
  
  // Create the DIV to hold the control and call the HomeControl() constructor  
  // passing in this DIV.  
  var homeControlDiv = document.createElement('div');  
  var homeControl = new HomeControl(homeControlDiv, map);  
  
  homeControlDiv.index = 1;  
  map.controls[google.maps.ControlPosition.TOP\_RIGHT].push(homeControlDiv);  
}

[Ver ejemplo (control-custom.html)](https://developers.google.com/maps/documentation/javascript/examples/control-custom?hl=es)

## Marcadores

Los marcadores identifican ubicaciones en el mapa. De manera predeterminada, utilizan un icono estándar, aunque puedes establecer un icono personalizado dentro del constructor del marcador o mediante la ejecución de setIcon() en el marcador. El constructor google.maps.Marker toma un único objeto literal *Marker options* que especifica las propiedades iniciales del marcador. A continuación se indican algunos campos especialmente importantes que se suelen definir al crear un marcador.

* position (obligatorio) especifica un valor de LatLng que identifica la ubicación inicial del marcador.
* map (opcional) especifica el objeto Map en el que se sitúa el marcador.

Ten en cuenta que debes especificar el mapa en el que vas a añadir el marcador dentro del constructor Marker. Si no especificas este argumento, el marcador se creará, pero no se añadirá al mapa (o no mostrará). Puedes añadir el marcador más tarde mediante la ejecución del método setMap() del marcador. Para eliminar un marcador, ejecuta el método setMap() y transmite null como el argumento.

Los marcadores están diseñados para ser interactivos. De forma predeterminada, reciben eventos 'click', por ejemplo, y se suelen utilizar dentro de detectores de eventos para abrir ventanas de información. Puedes establecer la propiedad draggable de un marcador en true para que [los usuarios puedan editar](https://developers.google.com/maps/documentation/javascript/overlays?hl=es#editable) el marcador en el mapa.

En el siguiente ejemplo, se añade un marcador simple a un mapa de Uluru, en el centro de Australia:

  var myLatlng = new google.maps.LatLng(-25.363882,131.044922);  
  var mapOptions = {  
    zoom: 4,  
    center: myLatlng,  
    mapTypeId: google.maps.MapTypeId.ROADMAP  
  }  
  var map = new google.maps.Map(document.getElementById("map\_canvas"), mapOptions);  
  
  var marker = new google.maps.Marker({  
      position: myLatlng,  
      map: map,  
      title:"Hello World!"  
  });

Este título de Marker se mostrará como información sobre la herramienta. ES DECIR COMO UN TOOL TIPS

Si no quieres incluir ninguna opción del marcador (*Marker options*) en el constructor del marcador, incluye un objeto {} vacío en el último argumento del constructor.

[Ver ejemplo (marker-simple.html)](https://developers.google.com/maps/documentation/javascript/examples/marker-simple?hl=es)

#### Animaciones

También puedes animar los marcadores de forma que muestren un movimiento dinámico en numerosas circunstancias diferentes. La propiedad animation, del tipogoogle.maps.Animation, especifica la forma en que está animado un marcador. Actualmente, se admiten los siguientes valores Animation:

* DROP indica que el marcador debe caer desde la parte superior del mapa hasta su ubicación definitiva en la que se colocase primero en el mapa. La animación finalizará una vez el marcador empiece a quedarse quieto y animation se restablezca a null. Este tipo de animación se suele especificar durante la creación del marcador (Marker).
* BOUNCE indica que el marcador debe "rebotar" en el mismo sitio. Un marcador que rebota seguirá haciéndolo hasta que la propiedad animation se defina explícitamente en null.

Puedes iniciar una animación en un marcador existente ejecutando setAnimation() en el objeto Marker.

El siguiente ejemplo crea un marcador en Estocolmo (Suecia) con una animación DROP. Al hacer clic en el marcador, se alterna entre una animación BOUNCE o ninguna animación:

var stockholm = new google.maps.LatLng(59.32522, 18.07002);  
var parliament = new google.maps.LatLng(59.327383, 18.06747);  
var marker;  
var map;  
  
function initialize() {  
  var mapOptions = {  
    zoom: 13,  
    mapTypeId: google.maps.MapTypeId.ROADMAP,  
    center: stockholm  
  };  
  
  map = new google.maps.Map(document.getElementById("map\_canvas"),  
      mapOptions);  
  
  marker = new google.maps.Marker({  
    map:map,  
    draggable:true,  
    animation: google.maps.Animation.DROP,  
    position: parliament  
  });  
  google.maps.event.addListener(marker, 'click', toggleBounce);  
}  
  
function toggleBounce() {  
  
  if (marker.getAnimation() != null) {  
    marker.setAnimation(null);  
  } else {  
    marker.setAnimation(google.maps.Animation.BOUNCE);  
  }  
}

[Ver ejemplo (marker-animations.html)](https://developers.google.com/maps/documentation/javascript/examples/marker-animations?hl=es)

### POLIGONOS

<!DOCTYPE html>  
<html>  
  <head>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <title>Circles</title>  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
    </style>  
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>  
    <script>  
// This example creates circles on the map, representing  
// populations in North America.  
  
// First, create an object containing LatLng and population for each city.  
var citymap = {};  
citymap['chicago'] = {  
  center: new google.maps.LatLng(41.878113, -87.629798),  
  population: 2714856  
};  
citymap['newyork'] = {  
  center: new google.maps.LatLng(40.714352, -74.005973),  
  population: 8405837  
};  
citymap['losangeles'] = {  
  center: new google.maps.LatLng(34.052234, -118.243684),  
  population: 3857799  
};  
citymap['vancouver'] = {  
  center: new google.maps.LatLng(49.25, -123.1),  
  population: 603502  
};  
  
var cityCircle;  
  
function initialize() {  
  // Create the map.  
  var mapOptions = {  
    zoom: 4,  
    center: new google.maps.LatLng(37.09024, -95.712891),  
    mapTypeId: google.maps.MapTypeId.TERRAIN  
  };  
  
  var map = new google.maps.Map(document.getElementById('map-canvas'),  
      mapOptions);  
  
  // Construct the circle for each value in citymap.  
  // Note: We scale the area of the circle based on the population.  
  for (var city in citymap) {  
    var populationOptions = {  
      strokeColor: '#FF0000',  
      strokeOpacity: 0.8,  
      strokeWeight: 2,  
      fillColor: '#FF0000',  
      fillOpacity: 0.35,  
      map: map,  
      center: citymap[city].center,  
      radius: Math.sqrt(citymap[city].population) \* 100  
    };  
    // Add the circle for this city to the map.  
    cityCircle = new google.maps.Circle(populationOptions);  
  }  
}  
  
google.maps.event.addDomListener(window, 'load', initialize);  
  
    </script>  
  </head>  
  <body>  
    <div id="map-canvas"></div>  
  </body>  
</html>

### INFO WINDOWS (INFORMACION DE MARCADORES)

<!DOCTYPE html>  
<html>  
  <head>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <title>Info windows</title>  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
    </style>  
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>  
    <script>  
// This example displays a marker at the center of Australia.  
// When the user clicks the marker, an info window opens.  
  
function initialize() {  
  var myLatlng = new google.maps.LatLng(-25.363882,131.044922);  
  var mapOptions = {  
    zoom: 4,  
    center: myLatlng  
  };  
  
  var map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);  
  
  var contentString = '<div id="content">'+  
      '<div id="siteNotice">'+  
      '</div>'+  
      '<h1 id="firstHeading" class="firstHeading">Uluru</h1>'+  
      '<div id="bodyContent">'+  
      '<p><b>Uluru</b>, also referred to as <b>Ayers Rock</b>, is a large ' +  
      'sandstone rock formation in the southern part of the '+  
      'Northern Territory, central Australia. It lies 335&#160;km (208&#160;mi) '+  
      'south west of the nearest large town, Alice Springs; 450&#160;km '+  
      '(280&#160;mi) by road. Kata Tjuta and Uluru are the two major '+  
      'features of the Uluru - Kata Tjuta National Park. Uluru is '+  
      'sacred to the Pitjantjatjara and Yankunytjatjara, the '+  
      'Aboriginal people of the area. It has many springs, waterholes, '+  
      'rock caves and ancient paintings. Uluru is listed as a World '+  
      'Heritage Site.</p>'+  
      '<p>Attribution: Uluru, <a href="http://en.wikipedia.org/w/index.php?title=Uluru&oldid=297882194">'+  
      'http://en.wikipedia.org/w/index.php?title=Uluru</a> '+  
      '(last visited June 22, 2009).</p>'+  
      '</div>'+  
      '</div>';  
  
  var infowindow = new google.maps.InfoWindow({  
      content: contentString  
  });  
  
  var marker = new google.maps.Marker({  
      position: myLatlng,  
      map: map,  
      title: 'Uluru (Ayers Rock)'  
  });  
  google.maps.event.addListener(marker, 'click', function() {  
    infowindow.open(map,marker);  
  });  
}  
  
google.maps.event.addDomListener(window, 'load', initialize);  
  
    </script>  
  </head>  
  <body>  
    <div id="map-canvas"></div>  
  </body>  
</html>

## CALCULO DE RUTA A TRAVES DE DOS PUNTOS

<!DOCTYPE html>  
<html>  
  <head>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <title>Directions service</title>  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
      #panel {  
        position: absolute;  
        top: 5px;  
        left: 50%;  
        margin-left: -180px;  
        z-index: 5;  
        background-color: #fff;  
        padding: 5px;  
        border: 1px solid #999;  
      }  
    </style>  
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>  
    <script>  
var directionsDisplay;  
var directionsService = new google.maps.DirectionsService();  
var map;  
  
function initialize() {  
  directionsDisplay = new google.maps.DirectionsRenderer();  
  var chicago = new google.maps.LatLng(41.850033, -87.6500523);  
  var mapOptions = {  
    zoom:7,  
    center: chicago  
  }  
  map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);  
  directionsDisplay.setMap(map);  
}  
  
function calcRoute() {  
  var start = document.getElementById('start').value;  
  var end = document.getElementById('end').value;  
  var request = {  
      origin:start,  
      destination:end,  
      travelMode: google.maps.TravelMode.DRIVING  
  };  
  directionsService.route(request, function(response, status) {  
    if (status == google.maps.DirectionsStatus.OK) {  
      directionsDisplay.setDirections(response);  
    }  
  });  
}  
  
google.maps.event.addDomListener(window, 'load', initialize);  
  
    </script>  
  </head>  
  <body>  
    <div id="panel">  
    <b>Start: </b>  
    <select id="start" onchange="calcRoute();">  
      <option value="chicago, il">Chicago</option>  
      <option value="st louis, mo">St Louis</option>  
      <option value="joplin, mo">Joplin, MO</option>  
      <option value="oklahoma city, ok">Oklahoma City</option>  
      <option value="amarillo, tx">Amarillo</option>  
      <option value="gallup, nm">Gallup, NM</option>  
      <option value="flagstaff, az">Flagstaff, AZ</option>  
      <option value="winona, az">Winona</option>  
      <option value="kingman, az">Kingman</option>  
      <option value="barstow, ca">Barstow</option>  
      <option value="san bernardino, ca">San Bernardino</option>  
      <option value="los angeles, ca">Los Angeles</option>  
    </select>  
    <b>End: </b>  
    <select id="end" onchange="calcRoute();">  
      <option value="chicago, il">Chicago</option>  
      <option value="st louis, mo">St Louis</option>  
      <option value="joplin, mo">Joplin, MO</option>  
      <option value="oklahoma city, ok">Oklahoma City</option>  
      <option value="amarillo, tx">Amarillo</option>  
      <option value="gallup, nm">Gallup, NM</option>  
      <option value="flagstaff, az">Flagstaff, AZ</option>  
      <option value="winona, az">Winona</option>  
      <option value="kingman, az">Kingman</option>  
      <option value="barstow, ca">Barstow</option>  
      <option value="san bernardino, ca">San Bernardino</option>  
      <option value="los angeles, ca">Los Angeles</option>  
    </select>  
    </div>  
    <div id="map-canvas"></div>  
  </body>  
</html>

[Ver ejemplo (directions-simple.html)](https://developers.google.com/maps/documentation/javascript/examples/directions-simple?hl=es)

## CALCULO DE RUTA CON INDICACIONES AL COSTADO

<!DOCTYPE html>  
<html>  
  <head>  
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">  
    <meta charset="utf-8">  
    <title>Displaying text directions with <code>setPanel()</code></title>  
    <style>  
      html, body, #map-canvas {  
        height: 100%;  
        margin: 0px;  
        padding: 0px  
      }  
      #panel {  
        position: absolute;  
        top: 5px;  
        left: 50%;  
        margin-left: -180px;  
        z-index: 5;  
        background-color: #fff;  
        padding: 5px;  
        border: 1px solid #999;  
      }  
    </style>  
    <style>  
      #directions-panel {  
        height: 100%;  
        float: right;  
        width: 390px;  
        overflow: auto;  
      }  
  
      #map-canvas {  
        margin-right: 400px;  
      }  
  
      #control {  
        background: #fff;  
        padding: 5px;  
        font-size: 14px;  
        font-family: Arial;  
        border: 1px solid #ccc;  
        box-shadow: 0 2px 2px rgba(33, 33, 33, 0.4);  
        display: none;  
      }  
  
      @media print {  
        #map-canvas {  
          height: 500px;  
          margin: 0;  
        }  
  
        #directions-panel {  
          float: none;  
          width: auto;  
        }  
      }  
    </style>  
    <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></script>  
    <script>  
var directionsDisplay;  
var directionsService = new google.maps.DirectionsService();  
  
function initialize() {  
  directionsDisplay = new google.maps.DirectionsRenderer();  
  var mapOptions = {  
    zoom: 7,  
    center: new google.maps.LatLng(41.850033, -87.6500523)  
  };  
  var map = new google.maps.Map(document.getElementById('map-canvas'),  
      mapOptions);  
  directionsDisplay.setMap(map);  
  directionsDisplay.setPanel(document.getElementById('directions-panel'));  
  
  var control = document.getElementById('control');  
  control.style.display = 'block';  
  map.controls[google.maps.ControlPosition.TOP\_CENTER].push(control);  
}  
  
function calcRoute() {  
  var start = document.getElementById('start').value;  
  var end = document.getElementById('end').value;  
  var request = {  
    origin: start,  
    destination: end,  
    travelMode: google.maps.TravelMode.DRIVING  
  };  
  directionsService.route(request, function(response, status) {  
    if (status == google.maps.DirectionsStatus.OK) {  
      directionsDisplay.setDirections(response);  
    }  
  });  
}  
  
google.maps.event.addDomListener(window, 'load', initialize);  
  
    </script>  
  </head>  
  <body>  
    <div id="control">  
      <strong>Start:</strong>  
      <select id="start" onchange="calcRoute();">  
        <option value="chicago, il">Chicago</option>  
        <option value="st louis, mo">St Louis</option>  
        <option value="joplin, mo">Joplin, MO</option>  
        <option value="oklahoma city, ok">Oklahoma City</option>  
        <option value="amarillo, tx">Amarillo</option>  
        <option value="gallup, nm">Gallup, NM</option>  
        <option value="flagstaff, az">Flagstaff, AZ</option>  
        <option value="winona, az">Winona</option>  
        <option value="kingman, az">Kingman</option>  
        <option value="barstow, ca">Barstow</option>  
        <option value="san bernardino, ca">San Bernardino</option>  
        <option value="los angeles, ca">Los Angeles</option>  
      </select>  
      <strong>End:</strong>  
      <select id="end" onchange="calcRoute();">  
        <option value="chicago, il">Chicago</option>  
        <option value="st louis, mo">St Louis</option>  
        <option value="joplin, mo">Joplin, MO</option>  
        <option value="oklahoma city, ok">Oklahoma City</option>  
        <option value="amarillo, tx">Amarillo</option>  
        <option value="gallup, nm">Gallup, NM</option>  
        <option value="flagstaff, az">Flagstaff, AZ</option>  
        <option value="winona, az">Winona</option>  
        <option value="kingman, az">Kingman</option>  
        <option value="barstow, ca">Barstow</option>  
        <option value="san bernardino, ca">San Bernardino</option>  
        <option value="los angeles, ca">Los Angeles</option>  
      </select>  
    </div>  
    <div id="directions-panel"></div>  
    <div id="map-canvas"></div>  
  </body>  
</html>

## EJEMPLOS EN LA PÁGINA OFICIAL

<https://developers.google.com/maps/documentation/javascript/examples/?hl=es>