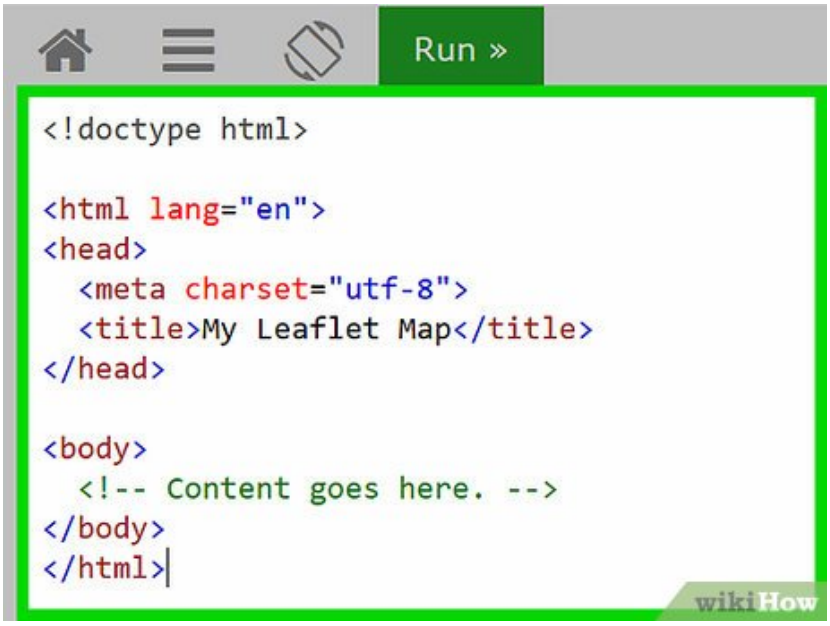
How to Make a Map Using Google Maps JS API

[https://developers.google.com/maps/documentation/javascript/ Google Maps JavaScript API] is an easy-to-use and fairly powerful JavaScript tool that enables us to create and display maps on a website. This article will guide you in setting...

Method 1 of 1:

Setting Up the Map

1. 

Open or create your webpage. If you don't already have a webpage you want to insert the map into, you can use the following HTML5 template; save it as 'map_page.html' :

```
html lang="en"> head> meta charset="utf-8"> title>My Leaflet Maptitle>
head> body>  body> html>
```

2. 

Include the Google Maps JS file. To do this, paste the following line of code into your HTML file, inside the area, on a new line, just before the closing tag:

```
script async defer src=
"https://maps.googleapis.com/maps/api/js?key=[YOUR_API_KEY]&callback=initMap
> script>
```

3. 

Get an API key and insert it into the JS URL. To use Google Maps JS API - just like any other Google API - you'll need an API key, which is like a passcode. You then need to insert your new key into the URL from the step above where it says [YOUR_API_KEY] (remove the square brackets).To get your own key, see Get a Key/Authentication page.

4.

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My Leaflet Map</title>
</head>
<body>
  <script async defer
  src="https://maps.googleapis.com/maps/api/js?key=
callback=initMap">
</script>
<div id="map"></div>
</body>
</html>
```


Add an HTML map container. Google Maps will display the map inside an HTML element, so you need to provide one for it. Paste the following line of markup inside the :

`div id="map">div>`

5.

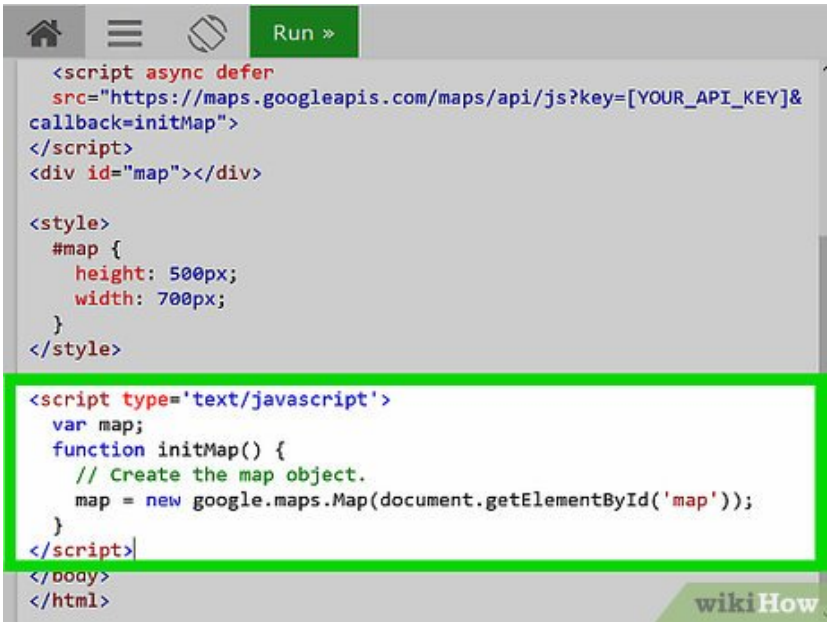
```
</head>
<body>
  <script async defer
  src="https://maps.googleapis.com/maps/api/js?key=
callback=initMap">
</script>
<div id="map"></div>

<style>
  #map {
    height: 500px;
    width: 700px;
  }
</style>
</body>
</html>
```


Style the map container. You need to set how large the map will be when it is displayed, and you should use CSS to do this. You can add the following to the document :

`style> #map { height: 500px; width: 700px; } style>`

6.



Create the map object. To start writing the JavaScript code that sets up the map, you'll need to add a area to the ***after*** the map container `div`. Also inside of this, you can create the map object by calling the `Map()` function of the google.maps object. You'll also need to pass the function the map container html element, like this:

```
script type='text/javascript'> var map; function initMap() {
// Create the map object. map = new google.maps.Map(document.
getElementById('map')); } /script>
```

7.

Set the map's zoom and center. The map's zoom and center properties determine the area that the map covers by default when the map is loaded. These are just the map's *default* values; the map's *current* zoom and center can be changed later if you enable those controls - we'll get to that later.
Note: You need to simply copy the following lines of code into your element (from the previous step),

and make sure it is ***inside*** the initMap() function, before the closing '}':

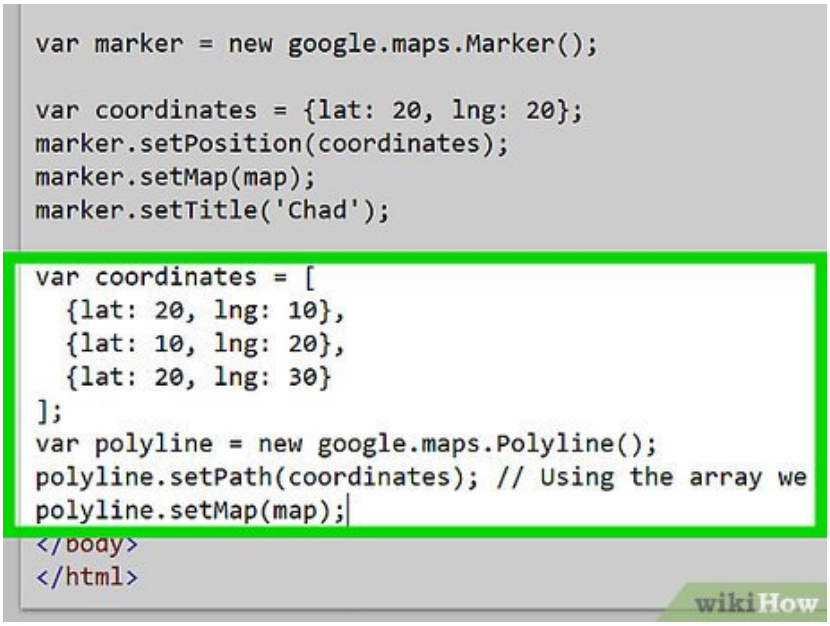
```
map.setZoom(2); map.setCenter({lat: 0, lng: 0});
```

8.

Add a marker. Markers show the location of a point on a map. In Leaflet, markers are a type of 'overlay', so they can be directly added to maps. By default, a marker appears as a red pin, which is a standard image that you can change.

```
var marker = new google.maps.Marker(); var coordinates = {lat: 20, lng:
 20}; marker.setPosition(coordinates); marker.setMap(map); marker.
setTitle('Chad');
```

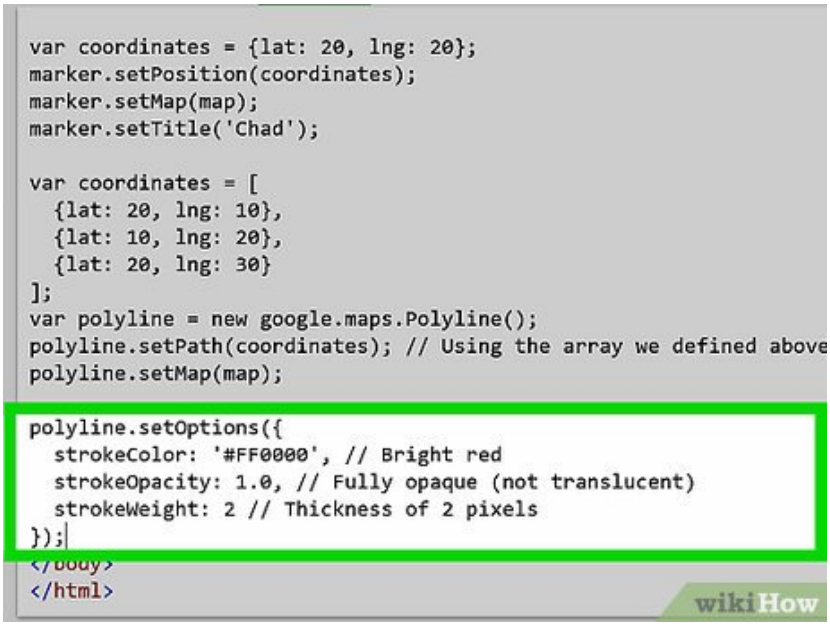
9.



Add polyline. Just like the marker, a polyline is a type of overlay, but it is used for a different purpose. A polyline is simply a line with multiple segments. You define it by providing the coordinates of the points that the line segments will be between. The coordinates needs to be an array of coordinate point objects. One of the great features of Google Maps' polylines is that they can be 'geodesic', which means they can curve to match the shape of the Earth! You can set that as an option later on.

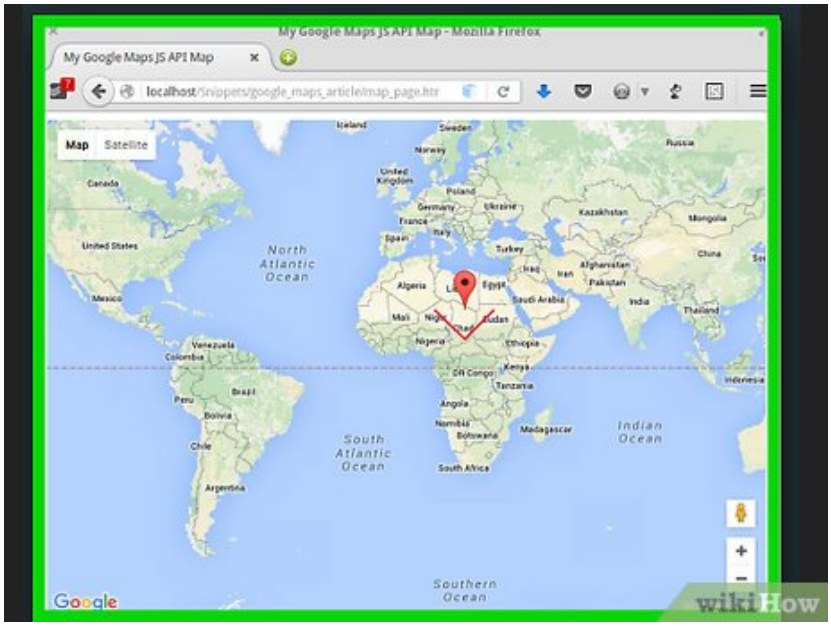
```
var coordinates = [ {lat: 20, lng: 10}, {lat: 10, lng: 20}, {lat: 20, lng: 30} ]; var polyline = new google.maps.Polyline(); polyline.setPath (coordinates); // Using the array we defined above polyline.setMap(map );
```

10.



Set options. Each overlay in Google Maps has options that you can set, that determine what the overlay looks like. You could set the options of the marker (from the steps above) to make it look unique, but below are options to set for the polyline you just created:

```
polyline.setOptions({ strokeColor: '#FF0000', // Bright red strokeOpacity: 1.0, // Fully opaque (not translucent) strokeWeight: 2 // Thickness of 2 pixels });
```

11. 

The finished map.

You finished reading the article "**How to Make a Map Using Google Maps JS API**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com