Google Maps API

2013

Tufail Soft Technology

Download free book at

tufailsoft.weebly.com

E-mail:
tufailsoft@hotmail.com

Mobile # 0334-4008671

Contents	Page No.

Contents	Page No.

BEGINNING GOOGLE MAPS API

Published by: Tufail Soft Technology

Face book: www.facebook.com/tufail.softtechnology

E-mail: tufailsoft@hotmail.com

Published on: 2013 (First Edition)

Chapter 1
INTRODUCTION

1.1 What is Google Map?

1.2 What is Google Map API?

1.3 What is API?

1.1 What is Google Map?

Google Maps is a web mapping service application and technology provided by Google that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit and maps embedded on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, car, bike etc. Google Maps satellite images are not updated in real time, but rather they are several months or years old.

1.2 What is Google Map API?

Google launched the Google Maps API in June 2005 to allow developers to integrate Google Maps into their websites. It is a free service.

By using the Google Maps API, it is possible to embed Google Maps site into an external website, on to which site specific data can be overlaid. Although initially only a JavaScript API, the Maps API was expanded to include an API for Adobe Flash applications. The Google Maps API is free for commercial use. The success of the Google Maps API has spawned a number of competing alternatives, including the Yahoo! Maps API, Bing Maps Platform, MapQuest Development Platform and OpenLayers.

In September 2011, Google announced it would discontinue a number of its products, including Google Maps API for Flash.

1.3 What is API?

API = Application programming interface.
An API is a specification used by software components to communicate with each other.
An API may describe the ways in which a particular task is performed.

Chapter 2

GOOGLE MAPS API KEY

Before you start, you will need a free API key from Google.

How to Get Started?

Before you start, you will need a specific API key from Google.

The key is free, and are required to complete this process.

Go to https://code.google.com/apis/console/, and log in with your Google Account.

The following will appear:

Click on the "Create Project" button.

In the list of services, find **Google Maps API v3,** and click on "off" to turn it on.

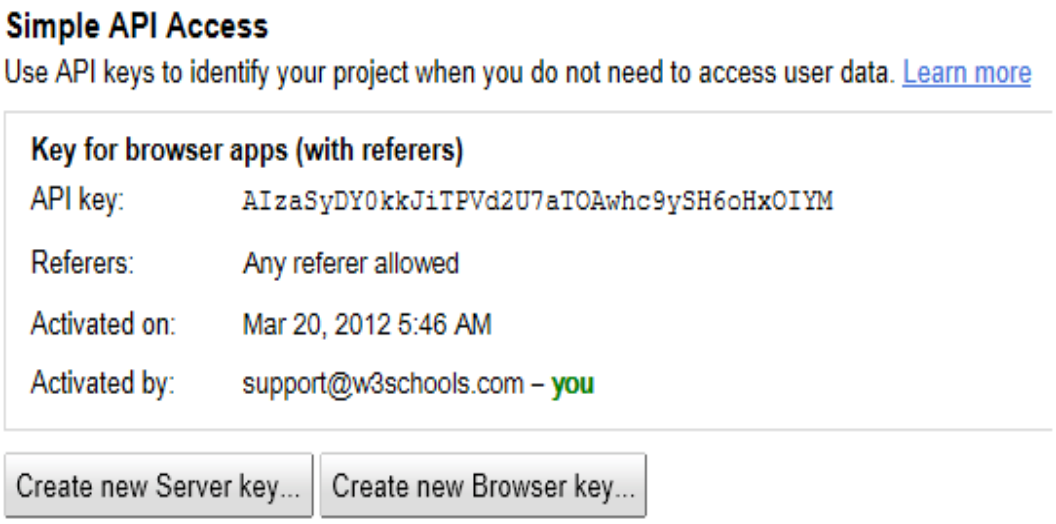
In the next screen, check "I Agree..." and the click the "Accept" button. You will now see that the button next to Google Maps API v3 has changed to "on".

Then click "API Access" in the menu to the left. It will ask you to "Create an OAuth 2.0 client id...".

In the next screen, provide a product name (e.g. "demo"), upload an image (if you want to) as a logo of your project, and click the "Next" button.

In the next screen, choose Application type ("Web application"), and type in your web address, and then click the "Create Client Id" button.

In the next screen, you have got an API key, and it will look something like this:



Note: Save your API key! (You will need it for all Google Maps applications you develop for the particular URL)

Chapter 3
Google Maps Basic

3.1 Create a Basic Google Map

3.2 Why Declare The Application as HTML5?

3.3 Add Google Maps API Key

3.4 Define Properties for the Map

3.5 Where to Show the Google Map

3.6 Create a Map Object

3.7 Loading the Map

3.1 Create a Basic Google Map

Now, we are ready to create a basic Google Map.
The example below creates a Google Map centered on Lahore, Pakistan:

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
function initialize()
{
var mapProp = { center:new google.maps.LatLng(30.508742,71.120850),
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };
var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:560px;height:480px;"></div>
</body>
</html>
```

Example Explained - Step By Step

The rest of this page explains the example above line by line in next topics.

3.2 Why Declare The Application as HTML5?

```
<!DOCTYPE html>
```

Most browsers will render pages with the HTML5 doctype in "standards mode" which means that your application is more cross-browser compliant.

Additionally, browsers that don't understand it will ignore it, and use "quirks mode" to display their content.

Tip: Be aware that some CSS that works within quirks mode is not valid in standards mode. In specific; all percentage-based sizes must inherit from parent block elements. If any of those ancestors fail to specify a size, they are set to 0 x 0 pixels. If you want to use percent, include the following <style> declaration:

```
<style type="text/css">
html {height:100%}
body {height:100%;margin:0;padding:0}
#googleMap {height:100%}
</style>
```

This style declaration indicates that the map container (googleMap) should take up 100% of the height of the HTML body.

3.3 Add Google Maps API Key

The first <script> tag in the example above is required, and it includes the Google Maps API:

```
<script
src="http://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=TRUE_OR_FALSE"
></script>
```

Insert your generated API key in the required **key** parameter (key=*YOUR_API_KEY)*.

The **sensor** parameter is required and it indicates whether this application uses a sensor (such as a GPS locator) to determine the user's location. Set this value to either true or false.

HTTPS

If your application is an HTTP Secure application, load the Google Maps API over HTTPS instead:

```
<script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=TRUE_OR_FALSE
"></script>
```

Asynchronously Loading

It is also possible to load the Google Maps API after the page has finished loading, or on demand.

The example below uses window.onload to load the Google Maps API after the page has fully loaded. The loadScript() function creates the Google Maps API <script> tag. In addition, the callback=initialize parameter is added to the end of the URL to only execute the initialize() function after the API is fully loaded:

```
function loadScript()
{
var script = document.createElement("script");
script.src = "http://maps.googleapis.com/maps/api/js?
key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false&callback=initialize"; document.body.appendChild(script);
}

window.onload = loadScript;
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script>
function initialize()
{
  var mapProp = {
    center: new google.maps.LatLng(30.508742,72.120850),
    zoom:7,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
function loadScript()
{
  var script = document.createElement("script");
  script.type = "text/javascript";
  script.src =
"http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false&callback=initialize";
```

```
  document.body.appendChild(script);
}
window.onload = loadScript;
</script>
</head>
<body>
<div id="googleMap" style="width:500px;height:500px;"></div>
</body>
```

3.4 Define Properties For The Map

To initialize a Map, we first create a Map properties object to define some properties for the map:

```
var mapProp = {
  center:new google.maps.LatLng(51.508742,-0.120850),
  zoom:7,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

Center

The center property specifies where to center the map. Create a LatLng object to center the map on a specific point. Pass the coordinates in the order: latitude, longitude.

Zoom

The zoom property specifies the initial zoom level for the map. zoom: 0 shows a map of the Earth fully zoomed out. Higher zoom levels zoom in at a higher resolution.

MapTypeId

The mapTypeId property specifies the initial map type to display.

The following map types are supported:

- ROADMAP (normal, default 2D map)
- SATELLITE (photographic map)
- HYBRID (photographic map + roads and city names)
- TERRAIN (map with mountains, rivers, etc.)

3.5 Where to Show the Google Map

A named element is often used to hold/show the Google Map:

```
<div id="googleMap" style="width:500px;height:380px;"></div>
```

Note: The map will always take its size from its containing element. So, always set a size on that element.

3.6 Create a Map Object

```
var map=new google.maps.Map(document.getElementById("googleMap")
,mapProp);
```

The code above creates a new map inside the element (googleMap) using the parameters that are passed (mapProp).

Tip: To create several maps on one page, just add new map objects.

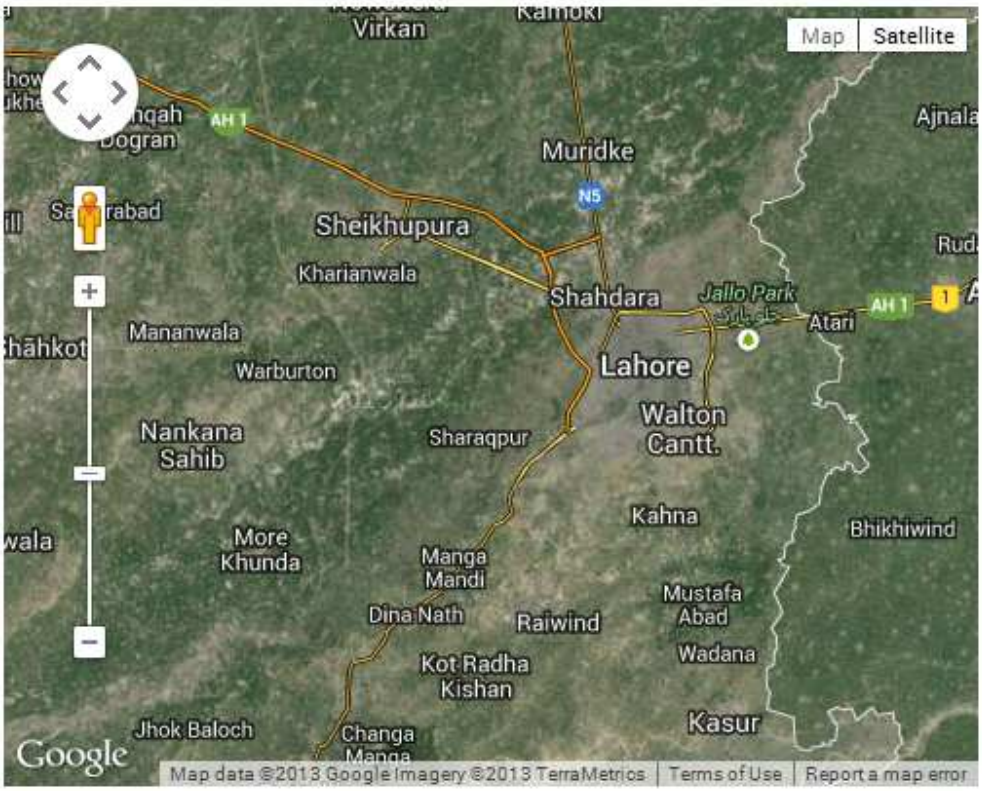
The example below defines four maps on one page (four maps with different map types):

```
var map = new
google.maps.Map(document.getElementById("googleMap"),mapProp);
var map2 = new
google.maps.Map(document.getElementById("googleMap2"),mapProp2);
var map3 = new
google.maps.Map(document.getElementById("googleMap3"),mapProp3);
var map4 = new
google.maps.Map(document.getElementById("googleMap4"),mapProp4);
```

Example:

```
</script>
<script>
function initialize()
{
 var mapProp = {
  center: new google.maps.LatLng(31.508742,74.120850),
  zoom:8,
  mapTypeId: google.maps.MapTypeId.ROADMAP
 };
 var mapProp2 = {
  center: new google.maps.LatLng(31.508742,74.120850),
  zoom:9,
  mapTypeId: google.maps.MapTypeId.SATELLITE
 };
 var mapProp3 = {
  center: new google.maps.LatLng(31.508742,74.120850),
  zoom:9,
  mapTypeId: google.maps.MapTypeId.HYBRID
 };
 var mapProp4 = {
  center: new google.maps.LatLng(31.508742,74.120850),
  zoom:9,
  mapTypeId: google.maps.MapTypeId.TERRAIN
 };
 var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
 var map2 = new
google.maps.Map(document.getElementById("googleMap2"),mapProp2);
 var map3 = new
google.maps.Map(document.getElementById("googleMap3"),mapProp3);
 var map4 = new
google.maps.Map(document.getElementById("googleMap4"),mapProp4);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>
<body>
```

```
<div id="googleMap" style="width:400px;height:300px;"></div>
<br>
<div id="googleMap2" style="width:400px;height:300px;"></div>
<br>
<div id="googleMap3" style="width:400px;height:300px;"></div>
<br>
<div id="googleMap4" style="width:400px;height:300px;"></div>
</body>
</html>
```


Map
Satellite
Virkan
Kamoki
Ajnala
Muridke
N5
Sheikhupura
Kharianwala
Shahdara
Jallo Park
AH 1
Atari
Mananwala
Warburton
Lahore
Walton Cantt.
Nankana Sahib
Sharaqpur
Kahna
Bhikhiwind
More Khunda
Manga Mandi
Mustafa Abad
Dina Nath
Raiwind
Wadana
Kot Radha Kishan
Jhok Baloch
Changa Manga
Kasur
Google
Map data ©2013 Google Imagery ©2013 TerraMetrics
Terms of Use
Report a map error


Map
Satellite
Virkan
Kamoki
Ajnala
Muridke
N5
Sheikhupura
Kharianwala
Shahdara
Jallo Park
AH 1
Atari
Mananwala
Warburton
Lahore
Nankana Sahib
Sharaqpur
Kahna
Bhikhiwind
More Khunda
Manga Mandi
Mustafa Abad
Dina Nath
Raiwind
Wadana
Kot Radha Kishan
Jhok Baloch
Changa Manga
Kasur
Google
Map data ©2013 Google
Terms of Use
Report a map error

3.7 Loading the Map

Execute the initialize() function that constructs the Map object on window load, to ensure that the map is placed on the page after the page is fully loaded:

```
google.maps.event.addDomListener(window, 'load', initialize);
```

Chapter 4

Google Maps Overlays

4.1 Google Maps –Overlays

4.2 Google Maps – Add a Marker

4.3 Google Maps- Animate the Marker

4.4 Google Maps- Icon Instead of Marker

4.5 Google Maps – Polyline

4.6 Google Maps-Polygon

4.7 Google Maps-Circle

4.8 Google Maps- InfoWindow

4.1 Google Maps – Overlays

Overlays are objects on the map that are bound to latitude/longitude coordinates.

Google Maps has several types of overlays:

- Marker - Single locations on a map. Markers can also display custom icon images
- Polyline - Series of straight lines on a map
- Polygon - Series of straight lines on a map, and the shape is "closed"
- Circle and Rectangle
- Info Windows - Displays content within a popup balloon on top of a map
- Custom overlays

4.2 Google Maps - Add a Marker

The Marker constructor creates a marker. (Note that the position property must be set for the marker to display).

Add the marker to the map by using the setMap() method:

```
var marker=new google.maps.Marker({
  position:myCenter,
  });

marker.setMap(map);
```

Exapmle:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
```

```
 center:myCenter,
 zoom:5,
 mapTypeId:google.maps.MapTypeId.ROADMAP
 };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

var marker=new google.maps.Marker({
 position:myCenter,
 });

marker.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

4.3 Google Maps - Animate the Marker

The example below shows how to animate the marker with the animation property:

```
marker=new google.maps.Marker({
position:myCenter,
animation:google.maps.Animation.BOUNCE
});

marker.setMap(map);
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);
var marker;

function initialize()
{
var mapProp = {
 center:myCenter,
 zoom:5,
 mapTypeId:google.maps.MapTypeId.ROADMAP
 };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

marker=new google.maps.Marker({
 position:myCenter,
 animation:google.maps.Animation.BOUNCE
 });

marker.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
```

```
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

4.4 Google Maps - Icon Instead of Marker

The example below specifies an image (icon) to use instead of the default marker:

```
var marker=new google.maps.Marker({
  position:myCenter,
  icon:'pinkball.png'
  });

marker.setMap(map);
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

var marker=new google.maps.Marker({
  position:myCenter,
  icon:'pinkball.png'
  });
```

```
marker.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

4.5 Google Maps – Polyline

A Polyline is a line that is drawn through a series of coordinates in an ordered sequence.

A polyline supports the following properties:

- path - specifies several latitude/longitude coordinates for the line
- strokeColor - specifies a hexadecimal color for the line (format: "#FFFFFF")
- strokeOpacity - specifies the opacity of the line (a value between 0.0 and 1.0)

- strokeWeight - specifies the weight of the line's stroke in pixels
- editable - defines whether the line is editable by users (true/false)

```
var myTrip = [stavanger,amsterdam,london];
var flightPath = new google.maps.Polyline({
  path:myTrip,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2
});
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var x=new google.maps.LatLng(31.508742,74.320850);
var rawalpindi=new google.maps.LatLng(33.462677,73.023906);
var lahore=new google.maps.LatLng(31.508742,74.320850);
var faisalabad=new google.maps.LatLng(31.450579,73.204089);
var multan=new google.maps.LatLng(30.092632,71.472635);


function initialize()
{
var mapProp = {
  center:x,
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

var myTrip=[rawalpindi,lahore,multan,faisalabad];
var flightPath=new google.maps.Polyline({
  path:myTrip,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2
```

```
  });

flightPath.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:600px;height:480px;"></div>
</body>
</html>
```

4.6 Google Maps – Polygon

A Polygon is similar to a Polyline in that it consists of a series of coordinates in an ordered sequence. However, polygons are designed to define regions within a closed loop.

Polygons are made of straight lines, and the shape is "closed" (all the lines connect up).

A polygon supports the following properties:

- path - specifies several LatLng coordinates for the line (first and last coordinate are equal)
- strokeColor - specifies a hexadecimal color for the line (format: "#FFFFFF")
- strokeOpacity - specifies the opacity of the line (a value between 0.0 and 1.0)
- strokeWeight - specifies the weight of the line's stroke in pixels
- fillColor - specifies a hexadecimal color for the area within the enclosed region (format: "#FFFFFF")
- fillOpacity - specifies the opacity of the fill color (a value between 0.0 and 1.0)
- editable - defines whether the line is editable by users (true/false)

```
var myTrip = [stavanger,amsterdam,london,stavanger];
var flightPath = new google.maps.Polygon({
  path:myTrip,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2,
  fillColor:"#0000FF",
  fillOpacity:0.4
});
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
```

```
var x=new google.maps.LatLng(31.508742,74.320850);
var rawalpindi=new google.maps.LatLng(33.462677,73.023906);
var lahore=new google.maps.LatLng(31.508742,74.320850);
var faisalabad=new google.maps.LatLng(31.450579,73.204089);
var multan=new google.maps.LatLng(30.092632,71.472635);

function initialize()
{
var mapProp = {
  center:x,
  zoom:5,
  mapTypeId: google.maps.MapTypeId.ROADMAP
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

var myTrip=[rawalpindi,lahore,multan,faisalabad];
var flightPath=new google.maps.Polygon({
  path:myTrip,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2,
  fillColor:"#0000FF",
  fillOpacity:0.4
  });

flightPath.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:480px;"></div>
</body>
</html>
```

4.7 Google Maps – Circle

A circle supports the following properties:

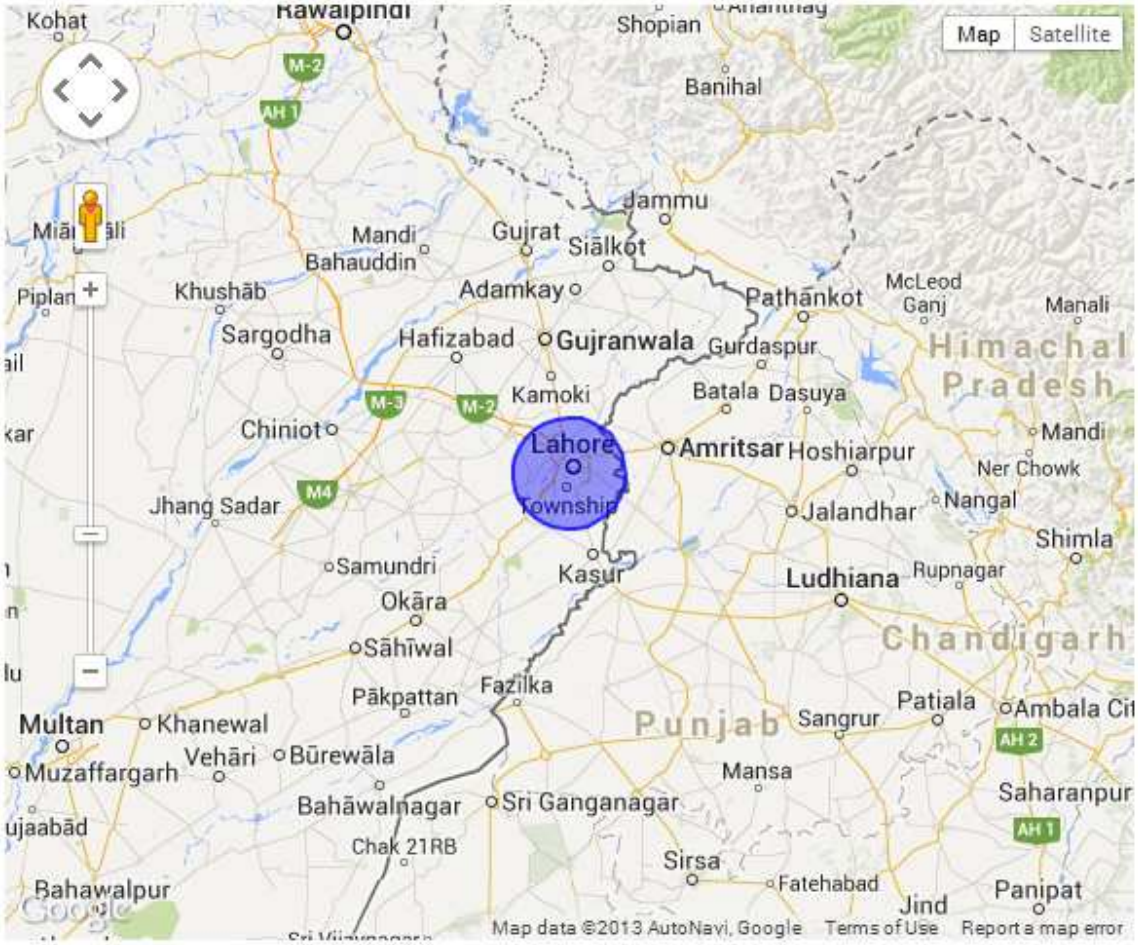
- center - specifies the google.maps.LatLng of the center of the circle
- radius - specifies the radius of the circle, in meters
- strokeColor - specifies a hexadecimal color for the line around the circle (format: "#FFFFFF")
- strokeOpacity - specifies the opacity of the stroke color (a value between 0.0 and 1.0)
- strokeWeight - specifies the weight of the line's stroke in pixels
- fillColor - specifies a hexadecimal color for the area within the circle (format: "#FFFFFF")
- fillOpacity - specifies the opacity of the fill color (a value between 0.0 and 1.0)
- editable - defines whether the circle is editable by users (true/false)

```
var myCity = new google.maps.Circle({
  center:amsterdam,
  radius:20000,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2,
  fillColor:"#0000FF",
  fillOpacity:0.4
});
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var amsterdam=new google.maps.LatLng(31.508742,74.320850);
function initialize()
{
var mapProp = {
  center:amsterdam,
  zoom:7,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);

var myCity = new google.maps.Circle({
  center:amsterdam,
  radius:30000,
  strokeColor:"#0000FF",
  strokeOpacity:0.8,
  strokeWeight:2,
  fillColor:"#0000FF",
  fillOpacity:0.4
  });

myCity.setMap(map);
}
```

```
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:580px;height:480px;"></div>
</body>
</html>
```



4.8 Google Maps - InfoWindow

Show an InfoWindow with some text content for a marker:

```
var infowindow = new google.maps.InfoWindow({
  content:"Hello World!"
  });

infowindow.open(map,marker);
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

var marker=new google.maps.Marker({
  position:myCenter,
  });

marker.setMap(map);

var infowindow = new google.maps.InfoWindow({
  content:"Lovely Lahore!"
  });

infowindow.open(map,marker);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:580px;height:480px;"></div>
</body>
</html>
```


Map
Satellite
Samarkand
Dushanbe
Tajikistan
Afghanistan
Kabul
Lovely Lahore!
Kashmir
Himachal Pradesh
Lahore
Faisalabad
Ludhiana
Pakistan
Multan
Punjab
Uttarakhand
Haryana
New Delhi
Uttar Pradesh
Nepal
Kathmandu
Jaipur
Rajasthan
Lucknow
Kanpur
Patna
Bihar
Karachi
Hyderabad
Madhya Pradesh
Jharkhand
Gujarat
Ahmedabad
India
West
Google
Map data ©2013 AutoNavi, Basarsoft, Google, MapIT, ORION-ME
Terms of Use

Chapter 5
Google Maps Events

5.1 Click The Marker to Zoom

5.2 Pan Back to Marker

5.3 Open an InfoWindow When Clicking on The Marker

5.4 Set Markers and Open Infowindow for Each Marker

5.1 Click The Marker to Zoom

We still use the map from the previous page: a map centered on London, England.

Now we want to zoom when a user is clicking on the marker (We attach an event handler to a marker that zooms the map when clicked).

Here is the added code:

```
// Zoom to 9 when clicking on marker
google.maps.event.addListener(marker,'click',function() {
  map.setZoom(9);
  map.setCenter(marker.getPosition());
  });
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
  center: myCenter,
  zoom:5,
  mapTypeId: google.maps.MapTypeId.ROADMAP
  };

var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);

var marker = new google.maps.Marker({
  position: myCenter,
  title:'Click to zoom'
  });

marker.setMap(map);
```

```
// Zoom to 9 when clicking on marker
google.maps.event.addListener(marker,'click',function() {
  map.setZoom(9);
  map.setCenter(marker.getPosition());
  });
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:580px;height:480px;"></div>

</body>
</html>
```

We register for event notifications using the addListener() event handler. That method takes an object, an event to listen for, and a function to call when the specified event occurs.

5.2 Pan Back to Marker

Here, we add an event handler to the map for changes to the 'center' property and pan the map back to the marker after 3 seconds on a center_changed event:

```
google.maps.event.addListener(map,'center_changed',function() {
  window.setTimeout(function() {
    map.panTo(marker.getPosition());
  },3000);
});
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
  center: myCenter,
  zoom:5,
  mapTypeId: google.maps.MapTypeId.ROADMAP
  };

var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);

var marker = new google.maps.Marker({
  position: myCenter,
  title:'Click to zoom'
  });

marker.setMap(map);

// Zoom to 9 when clicking on marker
google.maps.event.addListener(marker,'click',function() {
  map.setZoom(9);
  map.setCenter(marker.getPosition());
```

```
  });

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.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:580px;height:480px;"></div>

</body>
</html>
```

5.3 Open an InfoWindow When Clicking on The Marker

Click on the marker to show an infowindow with some text:

```
var infowindow = new google.maps.InfoWindow({
  content:"Hello World!"
  });

google.maps.event.addListener(marker, 'click', function() {
  infowindow.open(map,marker);
  });
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);

var marker=new google.maps.Marker({
  position:myCenter,
  });

marker.setMap(map);

var infowindow = new google.maps.InfoWindow({
  content:"Hello World!"
  });
```

```
google.maps.event.addListener(marker, 'click', function() {
  infowindow.open(map,marker);
  });
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

5.4 Set Markers and Open InfoWindow for Each Marker

Run a function when the user clicks on the map.

The placeMarker() function places a marker where the user has clicked, and shows an infowindow with the latitudes and longitudes of the marker:

```
google.maps.event.addListener(map, 'click', function(event) {
  placeMarker(event.latLng);
  });
```

```
function placeMarker(location) {
  var marker = new google.maps.Marker({
    position: location,
    map: map,
  });
  var infowindow = new google.maps.InfoWindow({
    content: 'Latitude: ' + location.lat() +
    '<br>Longitude: ' + location.lng()
  });
  infowindow.open(map,marker);
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var map;
var myCenter=new google.maps.LatLng(31.508742,74.320850);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

  map = new google.maps.Map(document.getElementById("googleMap"),mapProp);

  google.maps.event.addListener(map, 'click', function(event) {
    placeMarker(event.latLng);
  });
}

function placeMarker(location) {
  var marker = new google.maps.Marker({
    position: location,
```

```
    map: map,
  });
  var infowindow = new google.maps.InfoWindow({
    content: 'Latitude: ' + location.lat() + '<br>Longitude: ' + location.lng()
  });
  infowindow.open(map,marker);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>

</body>
</html>
```

Chapter 6
Google Maps Controls

6.1 Google Maps – The Default Controls

6.2 Google Maps – More Controls

6.3 Google Maps – Disabling The Default Controls

6.4 Google Maps – Turn On All Controls

6.5 Google Maps – Modifying Controls

6.6 Google Maps – Custom Controls

6.1 Google Maps - The Default Controls

When showing a standard Google map, it comes with the default control set:

- Zoom - displays a slider or "+/-" buttons to control the zoom level of the map
- Pan - displays a pan control for panning the map
- MapType - lets the user toggle between map types (roadmap and satellite)
- Street View - displays a Pegman icon which can be dragged to the map to enable Street View

6.2 Google Maps - More Controls

In addition to the default controls, Google Maps also has:

- Scale - displays a map scale element
- Rotate - displays a small circular icon which allows you to rotate maps
- Overview Map - displays a thumbnail overview map reflecting the current map viewport within a wider area

You can specify which controls to show when creating the map (inside MapOptions) or by calling setOptions() to change the map's options.

6.3 Google Maps - Disabling The Default Controls

You may instead wish to turn off the default controls.

To do so, set the Map's disableDefaultUI property (within the Map options object) to true:

disableDefaultUI:true

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>
```

```
<script>
function initialize()
{
  var mapProp = {
    center: new google.maps.LatLng(31.508742,74.320850),
    zoom:7,
    disableDefaultUI:true,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

6.4 Google Maps - Turn On All Controls

Some controls appear on the map by default; while others will not appear unless you set them.

Adding or removing controls from the map is specified in the Map options object.

Set the control to true to make it visible - Set the control to false to hide it.

The following example turns "on" all controls:

panControl:true,
zoomControl:true,
mapTypeControl:true,
scaleControl:true,
streetViewControl:true,
overviewMapControl:true,
rotateControl:true

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
function initialize()
{
 var mapProp = {
   center: new google.maps.LatLng(31.508742,74.320850),
   zoom:7,
   panControl:true,
   zoomControl:true,
   mapTypeControl:true,
   scaleControl:true,
   streetViewControl:true,
   overviewMapControl:true,
   rotateControl:true,
   mapTypeId: google.maps.MapTypeId.ROADMAP
 };
 var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>
```

```
<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

6.5 Google Maps - Modifying Controls

Several of the map controls are configurable.
The controls can be modified by specifying control options fields.

For example, options for modifying a Zoom control are specified in the zoomControlOptions field. The zoomControlOptions field may contain:

- google.maps.ZoomControlStyle.SMALL - displays a mini-zoom control (only + and - buttons)
- google.maps.ZoomControlStyle.LARGE - displays the standard zoom slider control
- google.maps.ZoomControlStyle.DEFAULT - picks the best zoom control based on device and map size

```
zoomControl:true,
zoomControlOptions: {
  style:google.maps.ZoomControlStyle.SMALL
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
function initialize()
{
  var mapProp = {
    center: new google.maps.LatLng(31.508742,74.320850),
    zoom:7,
    zoomControl:true,
    zoomControlOptions: {
      style:google.maps.ZoomControlStyle.SMALL
    },
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

Note: If you modify a control, always enable the control first (set it to true).

Another configurable control is the MapType control.

Options for modifying a control are specified in the mapTypeControlOptions field. The mapTypeControlOptions field may contain::

- google.maps.MapTypeControlStyle.HORIZONTAL_BAR - display one button for each map type
- google.maps.MapTypeControlStyle.DROPDOWN_MENU - select map type via a dropdown menu
- google.maps.MapTypeControlStyle.DEFAULT - displays the "default" behavior (depends on screen size)

```
mapTypeControl:true,
mapTypeControlOptions: {
  style:google.maps.MapTypeControlStyle.DROPDOWN_MENU
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
function initialize()
{
  var mapProp = {
    center: new google.maps.LatLng(31.508742,74.320850),
    zoom:7,
    mapTypeControl:true,
    mapTypeControlOptions: {
      style:google.maps.MapTypeControlStyle.DROPDOWN_MENU
    },
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
```

```
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

You can also position a control, with the ControlPosition property:

```
mapTypeControl:true,
mapTypeControlOptions: {
  style:google.maps.MapTypeControlStyle.DROPDOWN_MENU,
  position:google.maps.ControlPosition.TOP_CENTER
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
```

```
</script>

<script>
function initialize()
{
 var mapProp = {
   center: new google.maps.LatLng(31.508742,74.320850),
   zoom:7,
   mapTypeControl:true,
   mapTypeControlOptions: {
     style:google.maps.MapTypeControlStyle.DROPDOWN_MENU,
     position:google.maps.ControlPosition.TOP_CENTER
   },
   mapTypeId: google.maps.MapTypeId.ROADMAP
 };
 var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

6.6 Google Maps - Custom Controls

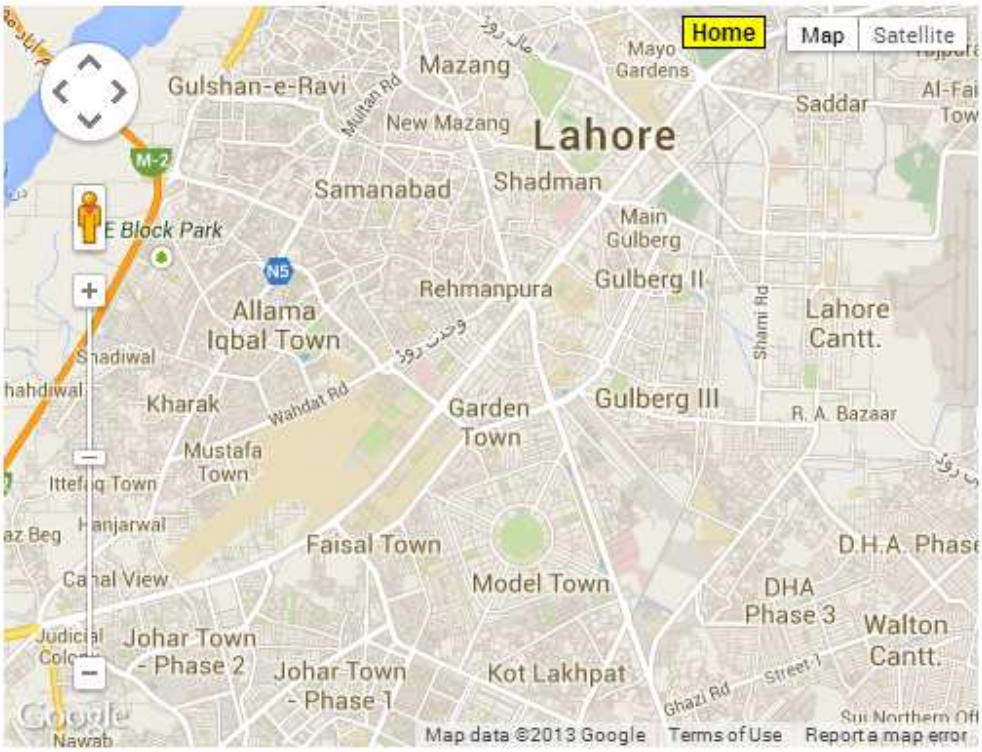
Create a custom control that always takes you back to London, when clicked (if the map is dragged):

```
controlDiv.style.padding = '5px';
var controlUI = document.createElement('div');
controlUI.style.backgroundColor = 'yellow';
controlUI.style.border='1px solid';
controlUI.style.cursor = 'pointer';
controlUI.style.textAlign = 'center';
controlUI.title = 'Set map to London';
controlDiv.appendChild(controlUI);
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 = '<b>Home<b>'
controlUI.appendChild(controlText);
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var map;
var lahore = new google.maps.LatLng(31.508742,74.320850);

// Add a Home control that returns the user to Lahore
function HomeControl(controlDiv, map) {
  controlDiv.style.padding = '5px';
  var controlUI = document.createElement('div');
  controlUI.style.backgroundColor = 'yellow';
  controlUI.style.border='1px solid';
  controlUI.style.cursor = 'pointer';
```

```
  controlUI.style.textAlign = 'center';
  controlUI.title = 'Set map to Lahore';
  controlDiv.appendChild(controlUI);
  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 = '<b>Home<b>'
  controlUI.appendChild(controlText);

  // Setup click-event listener: simply set the map to London
  google.maps.event.addDomListener(controlUI, 'click', function() {
    map.setCenter(lahore)
  });
}

function initialize() {
  var mapDiv = document.getElementById('googleMap');
  var myOptions = {
    zoom: 12,
    center: lahore,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  }
  map = new google.maps.Map(mapDiv, myOptions);
  // Create a DIV to hold the control and call HomeControl()
  var homeControlDiv = document.createElement('div');
  var homeControl = new HomeControl(homeControlDiv, map);
//  homeControlDiv.index = 1;
  map.controls[google.maps.ControlPosition.TOP_RIGHT].push(homeControlDiv);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```


Home
Map
Satellite
Gulshan-e-Ravi
Mazang
Mayo Gardens
Saddar
New Mazang
Lahore
Samanabad
Shadman
E Block Park
Main Gulberg
Gulberg II
Rehmanpura
Allama Iqbal Town
Lahore Cantt.
Kharak
Wahdat Rd
Garden Town
Gulberg III
R. A. Bazaar
Mustafa Town
Faisal Town
Model Town
D.H.A. Phase
Canal View
DHA Phase 3
Walton Cantt.
Johar Town - Phase 2
Johar Town - Phase 1
Kot Lakhpat
Ghazi Rd
Nawab
Map data ©2013 Google
Terms of Use
Report a map error

Chapter 7
Google Maps Types

7.1 Google Maps – Basic Map Types

7.2 Google Maps – 45° Perspective View

7.3 Google Maps – Disable 45° Perspective View – setTilt(0)

7.1 Google Maps - Basic Map Types

The following map types are supported in Google Maps API:

- ROADMAP (normal, default 2D map)
- SATELLITE (photographic map)
- HYBRID (photographic map + roads and city names)
- TERRAIN (map with mountains, rivers, etc.)

The map type is specified either within the Map properties object, with the mapTypeId property:

```
var mapProp = {
  center:new google.maps.LatLng(51.508742,-0.120850),
  zoom:7,
  mapTypeId: google.maps.MapTypeId.HYBRID
};
```

Or by calling the map's setMapTypeId() method:

```
map.setMapTypeId(google.maps.MapTypeId.HYBRID);
```

7.2 Google Maps - 45° Perspective View

The map types SATELLITE and HYBRID support a 45° perspective imagery view for certain locations (only at high zoom levels).

If you zoom into a location with 45° imagery view, the map will automatically alter the perspective view. In addition, the map will add:

- A compass wheel around the Pan control, allowing you to rotate the image
- A Rotate control between the Pan and Zoom controls, allowing you to rotate the image 90°
- A toggle control for displaying the 45° perspective view, under the Satellite control/label

Note: Zooming out from a map with 45° imagery will revert each of these changes, and the original map is displayed.

The following example shows a 45° perspective view of Palazzo Ducale in Venice, Italy:

```
var mapProp = {
center:myCenter,
zoom:18,
mapTypeId:google.maps.MapTypeId.HYBRID
};
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.311682,74.241608);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:18,
  mapTypeId:google.maps.MapTypeId.HYBRID
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:580px;height:480px;"></div>
</body>
</html>
```

Tip: Google adds 45° Imagery for new locations all the time. Check out the list of 45° imagery on Google Maps.

7.3 Google Maps - Disable 45° Perspective View - setTilt(0)

You can disable 45° perspective view by calling setTilt(0) on the Map object:

map.setTilt(0);

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
```

```
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
var myCenter=new google.maps.LatLng(31.311682,74.241608);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:18,
  mapTypeId:google.maps.MapTypeId.HYBRID
  };

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
map.setTilt(0);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

Tip: To enable 45° perspective view at a later point, call setTilt(45).

Chapter 8

Google Maps API Reference

8.1 Map

8.2 Overlays

8.3 Events

8.4 Controls

8.1 Map

Constructor/Object	Description
Map()	Creates a new map inside a specified HTML element (typically a element)

8.2 Overlays

Constructor/Object	Description
Marker	Creates a marker. (Note that the position must be set for the marker to display)
MarkerOptions	Options for rendering the marker
MarkerImage	A structure representing a Marker icon or shadow image
MarkerShape	Defines the marker shape to use in determination of a marker's clickable region (type and coord)
Animation	Specifies animations that can be played on a marker (bounce or drop)
InfoWindow	Creates an info window
InfoWindowOptions	Options for rendering the info window
Polyline	Creates a polyline (contains path and stroke styles)
PolylineOptions	Options for rendering the polyline
Polygon	Creates a polygon (contains path and stroke+fill styles)
PolygonOptions	Options for rendering the polygon
Rectangle	Creates a rectangle (contains bounds and stroke+fill styles)
RectangleOptions	Options for rendering the rectangle
Circle	Creates a circle (contains center+radius and stroke+fill styles)
CircleOptions	Options for rendering the circle
GroundOverlay	
GroundOverlayOptions	
OverlayView	
MapPanes	
MapCanvasProjection	

8.3 Events

Constructor/Object	Description
MapsEventListener	It has no methods and no constructor. Its instances are returned from addListener(), addDomListener() and are eventually passed back to removeListener()
event	Adds/Removes/Trigger event listeners
MouseEvent	Returned from various mouse events on the map and overlays

8.4 Controls

Constructor/Object	Description
MapTypeControlOptions	Holds options for modifying a control (position and style)
MapTypeControlStyle	Specifies what kind of map control to display (Drop-down menu or buttons)
OverviewMapControlOptions	Options for rendering of the overview map control (opened or collapsed)
PanControlOptions	Options for rendering of the pan control (position)
RotateControlOptions	Options for rendering of the rotate control (position)
ScaleControlOptions	Options for rendering of the scale control (position and style)
ScaleControlStyle	Specifies what kind of scale control to display
StreetViewControlOptions	Options for rendering of the street view pegman control (position)
ZoomControlOptions	Options for rendering of the zoom control (position and style)
ZoomControlStyle	Specifies what kind of zoom control to display (large or small)
ControlPosition	Specifies the placement of controls on the map

Chapter 9

Maps API Map() Constructor

9.1 Definition and Usage

9.2 Parameter Value

9.3 Methods of Map()

9.4 Properties of Map()

9.5 Events of Map()

Create a Google Map:

var map=new google.maps.Map(document.getElementById("googleMap"),mapOpt);

Example:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>

<script>
function initialize()
{
var mapOpt = { center:new google.maps.LatLng(31.508742,74.320850),  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };
var map=new google.maps.Map(document.getElementById("googleMap"),mapOpt);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

9.1 Definition and Usage

The Map() constructor creates a new map inside a specified HTML element (typically a element).

Syntax

```
new google.maps.Map(HTMLElement,MapOptions)
```

9.2 Parameter Values

Parameter	Description
HTMLElement	Specifies in what HTML element to put the map
MapOptions	A MapOptions object that holds the map initialization variables/options

9.3 Methods of Map()

Method	Return Value	Description
fitBounds(*LatLngBounds*)	None	Sets the viewport to contain the given bounds
getBounds()	*LatLng,LatLng*	Returns the south-west latitude/longitude and the north-east latitude/longitude of the current viewport
getCenter()	*LatLng*	Returns the lat/lng of the center of the map
getDiv()	*Node*	Returns a DOM object that contains the map
getHeading()	*number*	Returns the compass heading of aerial imagery (for SATELLITE and HYBRID map types)
getMapTypeId()	HYBRID ROADMAP SATELLITE TERRAIN	Returns the current map type
getProjection()	*Projection*	Returns the current Projection
getStreetView()	*StreetViewPanorama*	Returns the default StreetViewPanorama bound to the map
getTilt()	*number*	Returns the angle of incidence for aerial imagery in degrees (for SATELLITE and HYBRID map types)
getZoom()	*number*	Returns the current zoom level of the map

panBy(*xnumber,ynumber*)	None	Changes the center of the map by the given distance in pixels
panTo(*LatLng*)	None	Changes the center of the map to the given LatLng
panToBounds(*LatLngBounds*)	None	Pans the map by the minimum amount necessary to contain the given LatLngBounds
setCenter(*LatLng*)	None	
setHeading(*number*)	None	Sets the compass heading for aerial imagery measured in degrees from cardinal direction North
setMapTypeId(*MapTypeId*)	None	Changes the kind of map to display
setOptions(*MapOptions*)	None	
setStreetView (*StreetViewPanorama*)	None	Binds a StreetViewPanorama to the map
setTilt(*number*)	None	Sets the angle of incidence for aerial imagery in degrees (for SATELLITE and HYBRID map types)
setZoom(*number*)	None	

9.4 Properties of Map()

Property	Type	Description
controls	*Array.<MVCArray.<Node>>*	Additional controls to attach to the map
mapTypes	*MapTypeRegistry*	A registry of MapType instances by string ID
overlayMapTypes	*MVCArray.<MapType>*	Additional map types to overlay

9.5 Events of Map()

Event	Arguments	Description
bounds_changed	None	Fired when the viewport bounds have changed
center_changed	None	Fired when the map center property changes
click	*MouseEvent*	Fired when the user clicks on the map
dblclick	*MouseEvent*	Fired when the user double-clicks on the map
drag	None	Fired repeatedly while the user drags the map
dragend	None	Fired when the user stops dragging the map
dragstart	None	Fired when the user starts dragging the map
heading_changed	None	Fired when the map heading property changes
idle	None	Fired when the map becomes idle after panning or zooming
maptypeid_changed	None	Fired when the mapTypeId property changes
mousemove	*MouseEvent*	Fired whenever the user's mouse moves over the map container
mouseout	*MouseEvent*	Fired when the user's mouse exits the map container
mouseover	*MouseEvent*	Fired when the user's mouse enters the map container
projection_changed	None	Fired when the projection has changed
resize	None	Fired when the map (div) changes size
rightclick	*MouseEvent*	Fired when the user right-clicks on the map
tilesloaded	None	Fired when the visible tiles have finished loading
tilt_changed	None	Fired when the map tilt property changes
zoom_changed	None	Fired when the map zoom property changes

Chapter 10

Latitude and Longitude

10.1 Punjab

10.2 Khyber Pukhtoon Khwa (KPK)

10.3 Sind & Baluchistan

10.1 Punjab

City Name	Latitude	Longitude
Rawalpindi	33.362677	73.023906
Sargodha	32.044019	72.402052
Gujranwala	32.085411	74.105814
Lahore	31.323447	74.203249
Faisalabad	31.250579	73.044089
Sahiwal	30.395413	73.063029
Multan	30.112632	71.272635
Bahawalpur	29.234765	71.405971
D.G.Khan	30.031337	70.381404

10.2 Khyber Pukhtoon Khwa (KPK)

City Name	Latitude	Longitude
Malakand	34.303188	71.542370
Hazara	32.474670	74.170230
Mardan	34.115450	72.024158
Peshawar	34.001588	71.324139
Kohat	33.350211	71.260225
Dera Ismail Khan	31.492686	70.543794

10.3 Sind & Baluchistan

City Name	Latitude	Longitude
Karachi	24.535970	67.015699
Hyderabad	25.225625	68.221052
Jacobabad	28.163930	68.270592
Quetta	30.124271	67.005618
Sibbi	29.325245	67.521791