Chapter 16

How to use the API for Google Maps

Objectives

Applied

1. Use the API for Google Maps to add a map to a web page that includes markers and messages.
2. Use the API for Google Maps to provide driving directions between two points on a map.

Knowledge

1. Describe the use of the Google Maps API in terms of classes, constructors, objects, methods, and properties.
2. Describe the geocoding feature of Google Maps.
3. Describe the use of listeners with the Google Maps API.
4. Describe the way that driving directions are provided with the Google Maps API.

Google URLs

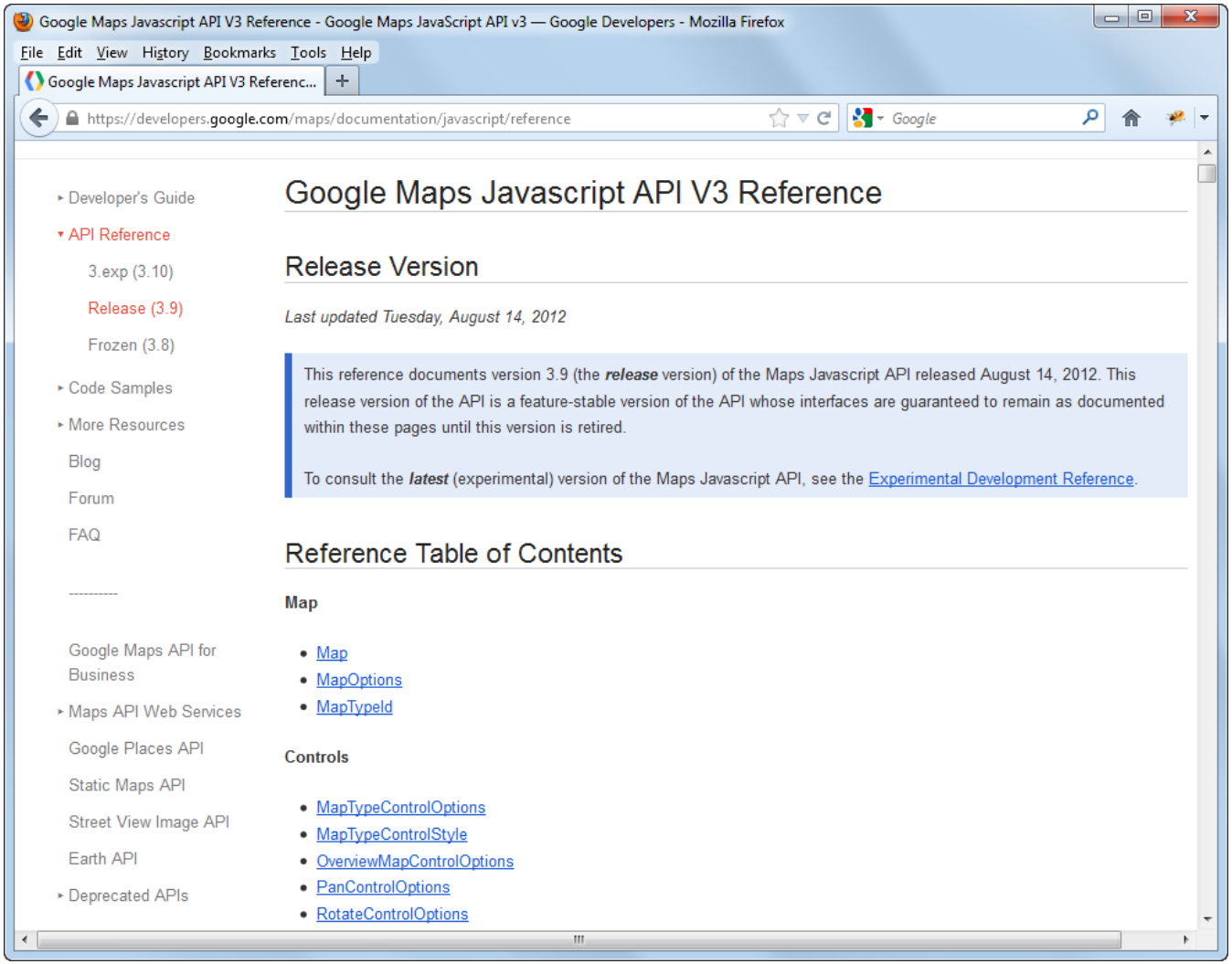
The URL for the Google APIs Services Console

```
code.google.com/apis/console/
```

The base URL for the Google Maps API documentation (on one line)

```
developers.google.com/maps/documentation/javascript/
reference
```

The web page for the start of the API reference



How to get a key for the Google Maps API

1. Go to the URL for the APIs services console that's shown above, and log in.
2. If this is the first time you've used the Google APIs, you'll be redirected to a welcome screen where you can click a "Create project" button. Then, click that button to go to the Console page.
3. Click on the Services link in the left sidebar.
4. Scroll down and enable the API v3 by changing its switch from "off" to "on".

The classes for adding a map to a web page

```
Map(element, options)
LatLng(latitude, longitude)
MapTypeId
```

The base URL for the Google Maps API reference (on one line)

```
developers.google.com/maps/documentation/javascript/
reference
```

The placeholders for specific classes

```
#Map
#MapTypeId
#LatLng
```

The required options for the Map constructor

Option	Setting
`zoom`	A number that provides the initial zoom level.
`center`	A new LatLng object that identifies the center of the map.
`mapTypeId`	A MapTypeId constant that specifies the map type.

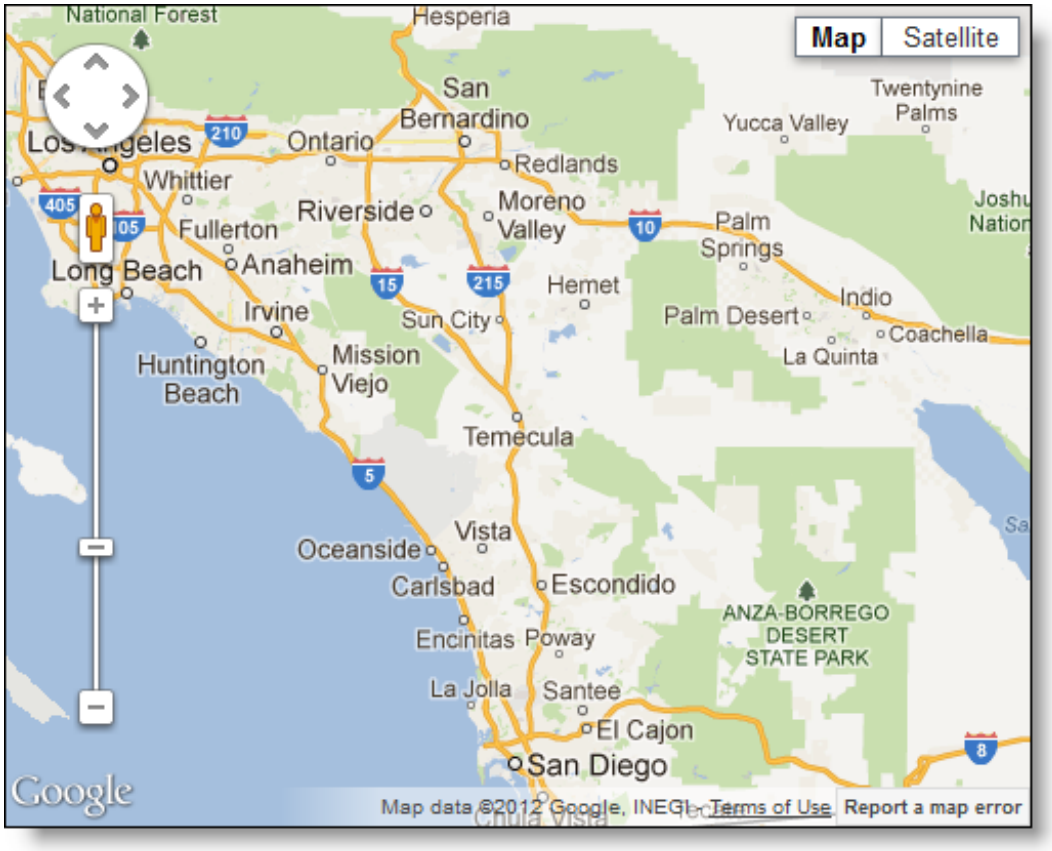
How to create a map object

```
var mapOptions = {
    zoom: 8,
    center:
        new google.maps.LatLng(33.4936111, -117.1475),
    mapTypeId: google.maps.MapTypeId.ROADMAP
};
var map = new google.maps.Map($("#map").get(0),
          mapOptions);
```

Terms

- constructor
- class
- object
- method
- property

A web page that displays a Google map



The HTML for the div element that will get the map

```
<div id="map"></div>
```

The script element for the Google Maps API

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

The jQuery for creating the map

```
$(document).ready(function(){
    var mapOptions = {
        zoom: 8,
        center:
           new google.maps.LatLng(33.4936111, -117.1475),
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map =
       new google.maps.Map($("#map").get(0), mapOptions);
});
```

Members of the Geocoder class

One of the methods of the Geocoder class

`geocode(`*`request, callback`*`)`

Two of the properties of the GeocoderRequest object

`address`

`location`

Three of the properties of the GeocoderResult object

`address_components`

`formatted_address`

`geometry.location`

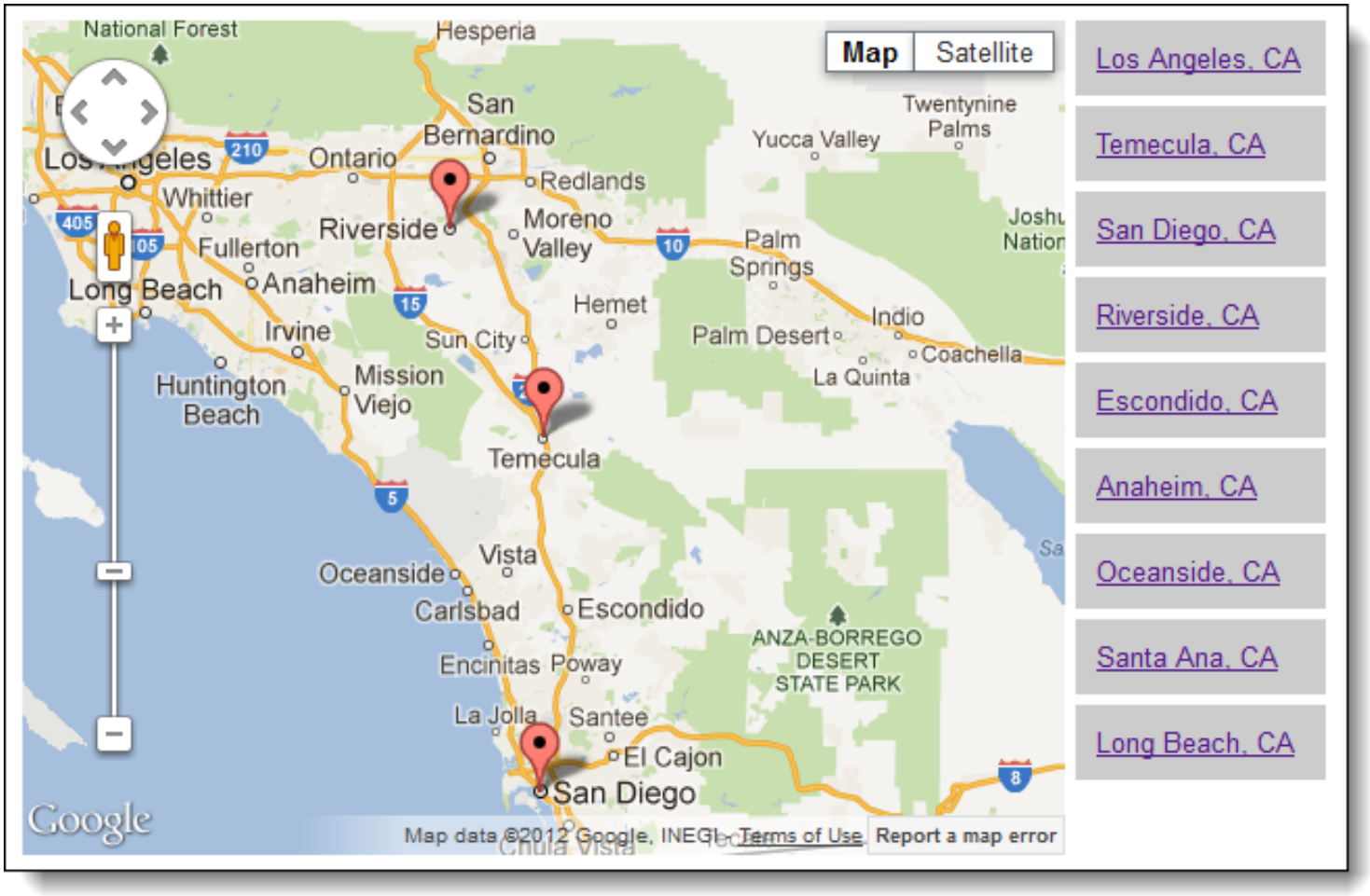
How to use the geocode method

```
geocoder = new google.maps.Geocoder();
geocoder.geocode({address: "4340 N Knoll, Fresno, CA"},
    function(results) {
        alert("Latitude for " +
            results[0].formatted_address + " is " +
            results[0].geometry.location.lat());
    });
```

How to create a Marker object

```
var marker = new google.maps.Marker({
    position:
        new google.maps.LatLng(36.799793, -119.858135),
    map: map});
// this assumes the map object is in a variable named map
```

A marker is added to the map when the user clicks on an address link



The HTML for the map and links

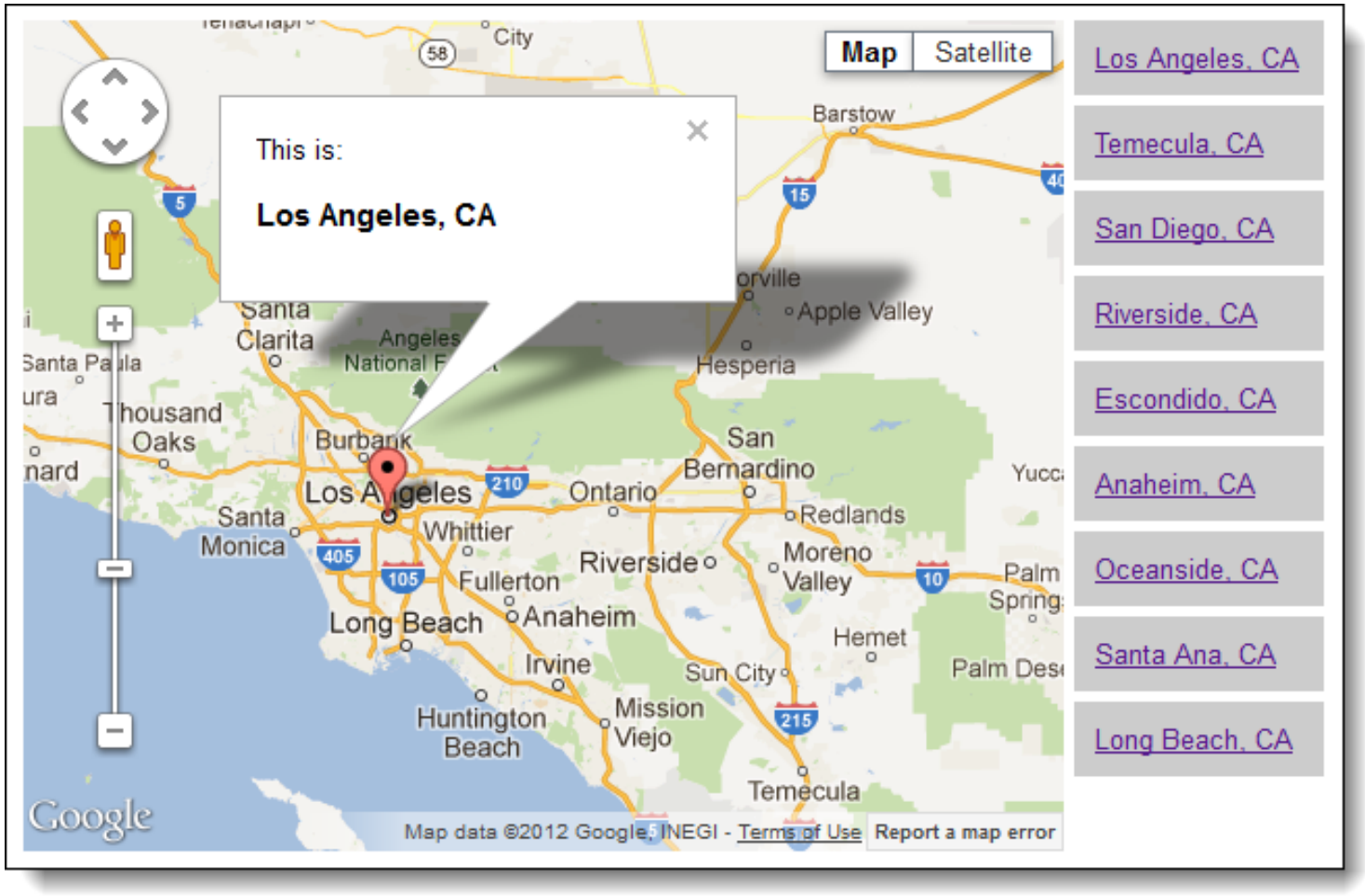
```
<div id="map"></div>
<ul id="links">
    <li><a href="#">Los Angeles, CA</a></li>
    <li><a href="#">Temecula, CA</a></li>
    <!-- the li elements for other cities -->
</ul>
```

The jQuery for the map and markers

```
$(document).ready(function(){
    var geocoder = new google.maps.Geocoder();
    var myLatLng =
        new google.maps.LatLng(33.4936111, -117.1475);
    var mapOptions = {zoom: 8, center: myLatLng,
        mapTypeId: google.maps.MapTypeId.ROADMAP};
    var map = new google.maps.Map($("#map").get(0),
        mapOptions);

    $("#links a").click(function() {
        // gets address from <a> element
        var address = $(this).text();
        geocoder.geocode({address: address},
            function(results) {
                new google.maps.Marker({
                    position: results[0].geometry.location,
                    map: map
            });
        });
    });
});
```

A marker and message ballon are added when the user clicks on a link



Members of the InfoWindow and Marker classes

One method of the InfoWindow class

open(map, marker)	Opens the message balloon for the map and marker.

Two methods of the Marker class

getPosition()	Gets the latitude and longitude coordinates of the marker.
setMap(*map*)	Renders the marker on the specified map. If the parameter is null, it removes the marker from the map.

The jQuery for displaying the message balloon

```
$(document).ready(function(){
    var marker;
    var geocoder = new google.maps.Geocoder();
    var myLatlng =
        new google.maps.LatLng(33.4936111, -117.1475);
    var mapOptions = {zoom: 8, center: myLatlng,
        mapTypeId: google.maps.MapTypeId.ROADMAP};
    var map = new google.maps.Map($("#map").get(0),
        mapOptions);

    $("#links a").click(function() {
        // get address from <a> element
        var address = $(this).text();
        // delete current marker (if any)
        if (marker) { marker.setMap(null); }
```

The jQuery (continued)

```
        geocoder.geocode({address: address},
            function(results) {
                marker = new google.maps.Marker({
                    position: results[0].geometry.location,
                    map: map
                });

                // add message balloon
                var infoWindow =
                    new google.maps.InfoWindow({
                        content:
                            "This is: <h3>" + address + "</h3>"}
                    );
                infoWindow.open(map, marker);
        });
    });
});
```

The classes for adding custom messages

`OverlayView()`	This constructor creates an object that can be used to overlay objects on the map.
`MapCanvasProjection`	An object is created from this class by the getProjection method of an OverlayView object.

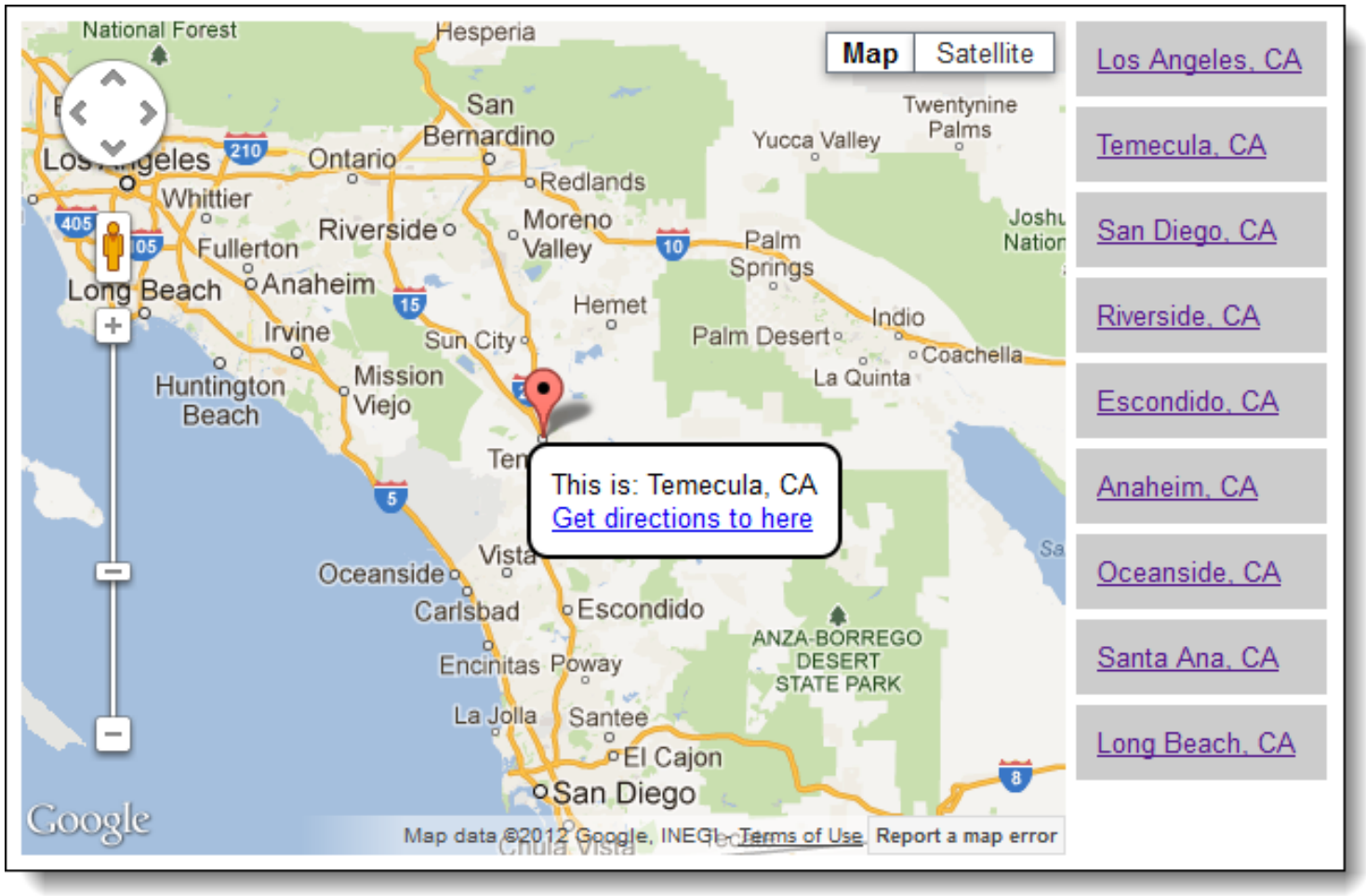
Three methods of the OverlayView class

`draw()`	Draws or updates the overlay.
`setMap(`*`map`*`)`	Adds the overlay to the map.
`getProjection()`	Returns a MapCanvasProjection object.

One method of the MapCanvasProjection class

`fromLatLngToContainerPixel(`*`markerPosition`*`)`	Returns the pixel coordinates for a marker's position.

A web page that displays a custom message for a marker



The HTML for the message balloon

```
<div id="message" style="display:none;"></div>
```

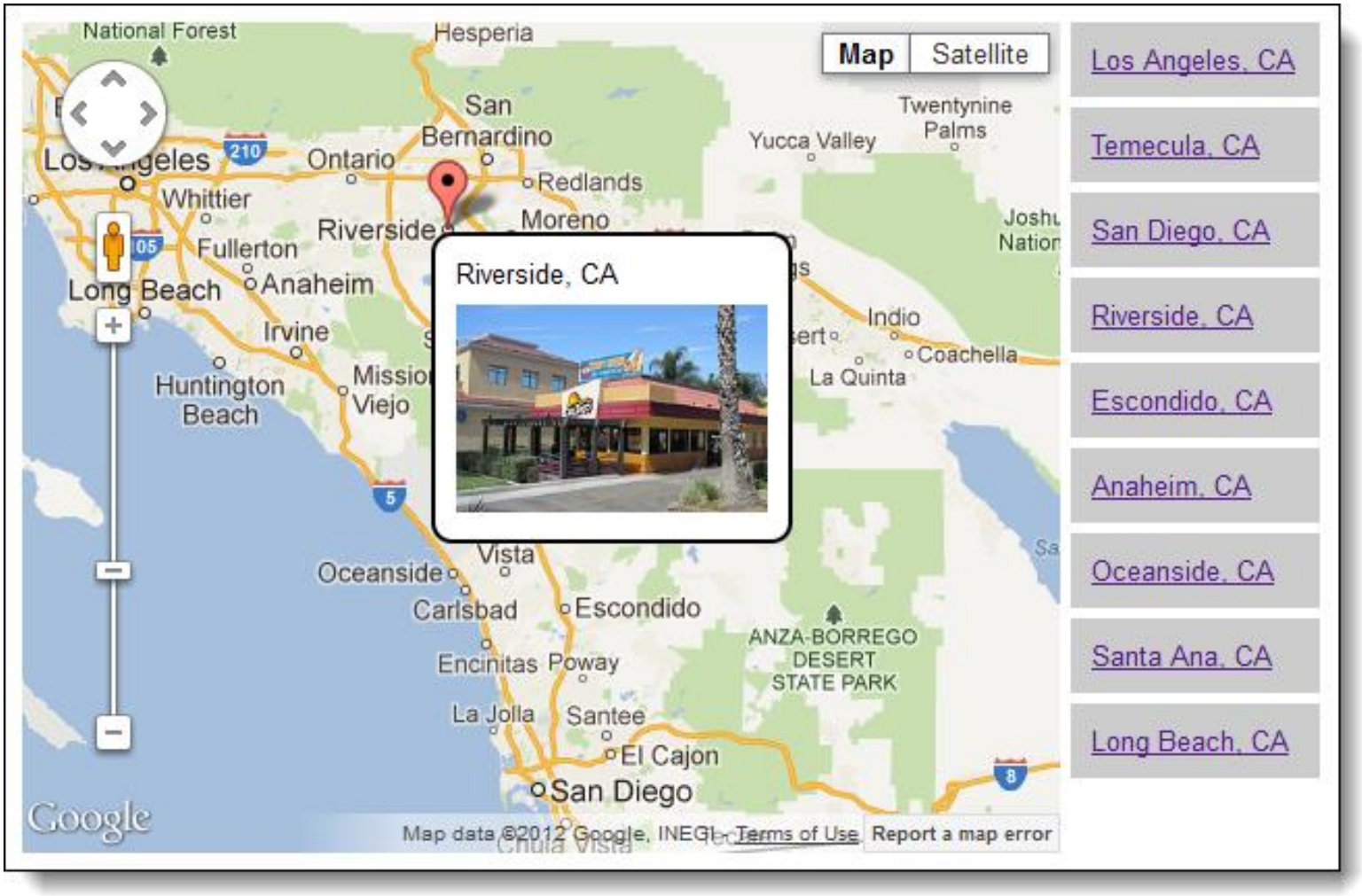
The jQuery for the custom messages

```
$(document).ready(function(){
    // same first 5 statements as before
    $("#links a").click(function() {
        var address = $(this).text();
        if (marker) { marker.setMap(null); }
        geocoder.geocode({address: address},
            function(results) {
            marker = new google.maps.Marker({
                position: results[0].geometry.location,
                map: map
            });
            var overlay = new google.maps.OverlayView();
            overlay.draw = function() {
                var point =
                    overlay.getProjection()
                    .fromLatLngToContainerPixel(
                        marker.getPosition());
```

The jQuery (continued)

```
                $("#message").html(
                  "This is: " + address + "<br>" +
                  "<a href=http://maps.google.com/maps? " +
                  "daddr=" + address +
                  ">Get directions to here</a>");
                $("#message").show().css({
                    top: point.y + 10,
                    left: point.x
                });
              };
              overlay.setMap(map);
        });
    });
});
```

A web page with a Flickr image for the location



The jQuery for setting up the click event handlers

```
// the same code as before
$("#links a").click(function() {
    var address = $(this).text();
    if (marker) { marker.setMap(null); }
    geocoder.geocode({address: address},
        function(results) {
            marker = new google.maps.Marker({
                position: results[0].geometry.location,
                map: map
            });
            var overlay = new google.maps.OverlayView();
            overlay.draw = function() {
                var point =
                    overlay.getProjection()
                    .fromLatLngToContainerPixel(
                        marker.getPosition());
```

The jQuery (continued)

```
            // the changed code for this application
            var url = "http://api.flickr.com/services/" +
                      "feeds/photos_public.gne?" +
                      "format=json&jsoncallback=?&tags=" +
                      address;
            $.getJSON(url, function(data) {
                $.each(data.items, function(i, item){
                    $("#message").html(address +
                        "<br><img src=" +
                        item.media.m +
                        " width=150 height=100>");
                });
            });
            $("#message").show().css({
                top: point.y + 10,
                left: point.x
            });
        };
        overlay.setMap(map);
    });
});
```

The classes for getting and displaying directions

Class	Description
`DirectionsService()`	This constructor creates an object that can be used to retrieve driving directions from the Google server.
`DirectionsRenderer()`	This constructor creates an object that can be used to render driving directions.
`TravelMode`	This class provides contants for four travel modes: DRIVING, BICYCLING, TRANSIT, and WALKING.

One method of the DirectionsService class

Member	Description
`route(request, callback)`	The first parameter is a DirectionsRequest object that has three required properties: origin (starting location), destination, and travel mode. This method returns a DirectionsResult object and a DirectionsStatus object.

Three members of the DirectionsRenderer class

Member	Description
`setMap(map)`	Specifies the map for which directions will be rendered.
`setPanel(panel)`	Specifies the HTML div element that will receive the directions.
`setDirections(result)`	Renders the result of a directions request in a div element.

A typical directions request

```
var request = {
    origin: marker1.getPosition(),
    destination: marker2.getPosition(),
    travelMode: google.maps.TravelMode.DRIVING
};
directionsService.route(request,
    function(result, status) {
        directionsRenderer.setDirections(result);
});
```

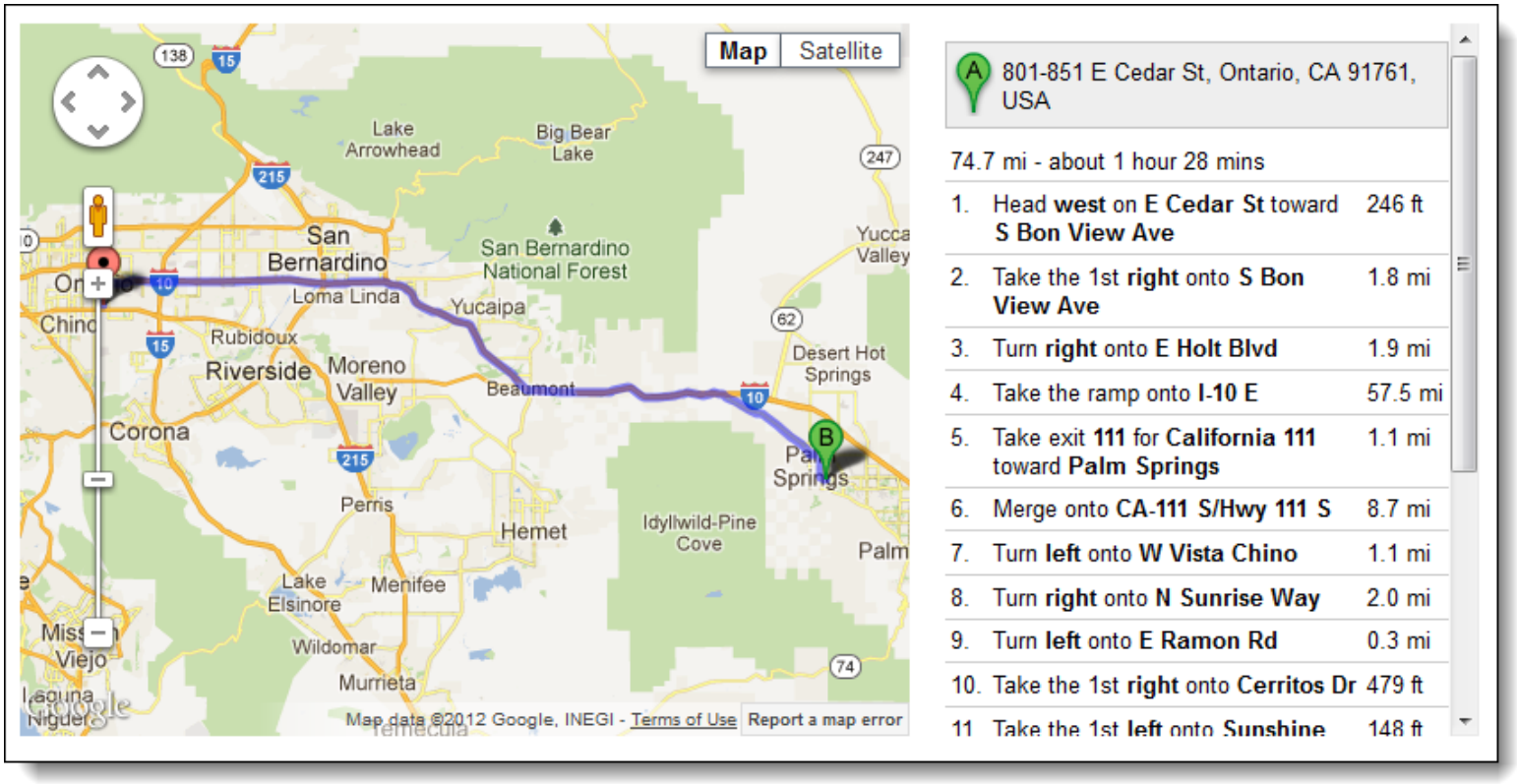
Three methods of the event namespace for using event handlers

Method	Description
`addListener(object, event, handler)`	Registers an event handler for an object and event. It returns an event object that has properties related to the event.
`removeListener(listener)`	Removes the specified listener.
`trigger(object, event)`	Fires the specified event for the specified object.

How to use the addListener method

```
google.maps.event.addListener(marker, "click",
    function(event) {
        // code for the event handler
});
```

A web page that displays driving directions



The HTML for the map and directions panel

```
<div id="map"></div>
<div id="directions"></div>
```

The jQuery for displaying the directions

```
// the code in the ready event handler
var directionsService =
    new google.maps.DirectionsService();
var markers = [];
var myLatlng =
    new google.maps.LatLng(33.4936111, -117.1475);
var mapOptions =
    {zoom: 8,
     center: myLatlng,
     mapTypeId: google.maps.MapTypeId.ROADMAP};
var map =
    new google.maps.Map($("#map").get(0), mapOptions);
```

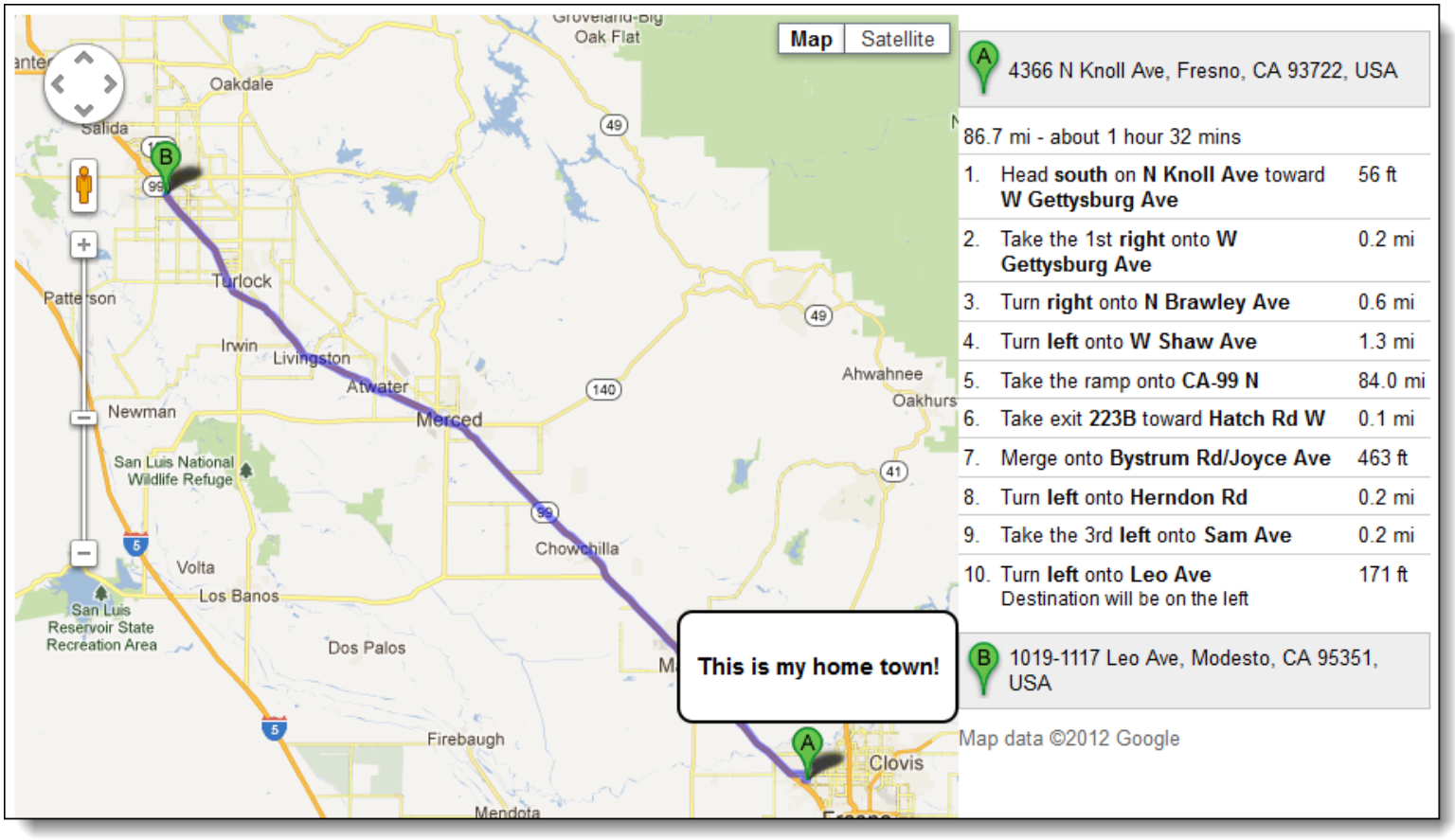
The jQuery (continued)

```
// the listener for the click event of the map
var listener = google.maps.event.addListener(map, "click",
    function(event) {
        var marker =
            new google.maps.Marker({
                position: event.latLng, map: map});
        markers.push(marker);
        if (markers.length > 1) {
            google.maps.event.removeListener(listener);
        var marker1 = markers[0];
        var marker2 = markers[1];
        var directionsRenderer =
            new google.maps.DirectionsRenderer();
        directionsRenderer.setMap(map);
        directionsRenderer.setPanel($("#directions")
            .get(0));
        var request = {
            origin: marker1.getPosition(),
            destination: marker2.getPosition(),
            travelMode: google.maps.TravelMode.DRIVING };
```

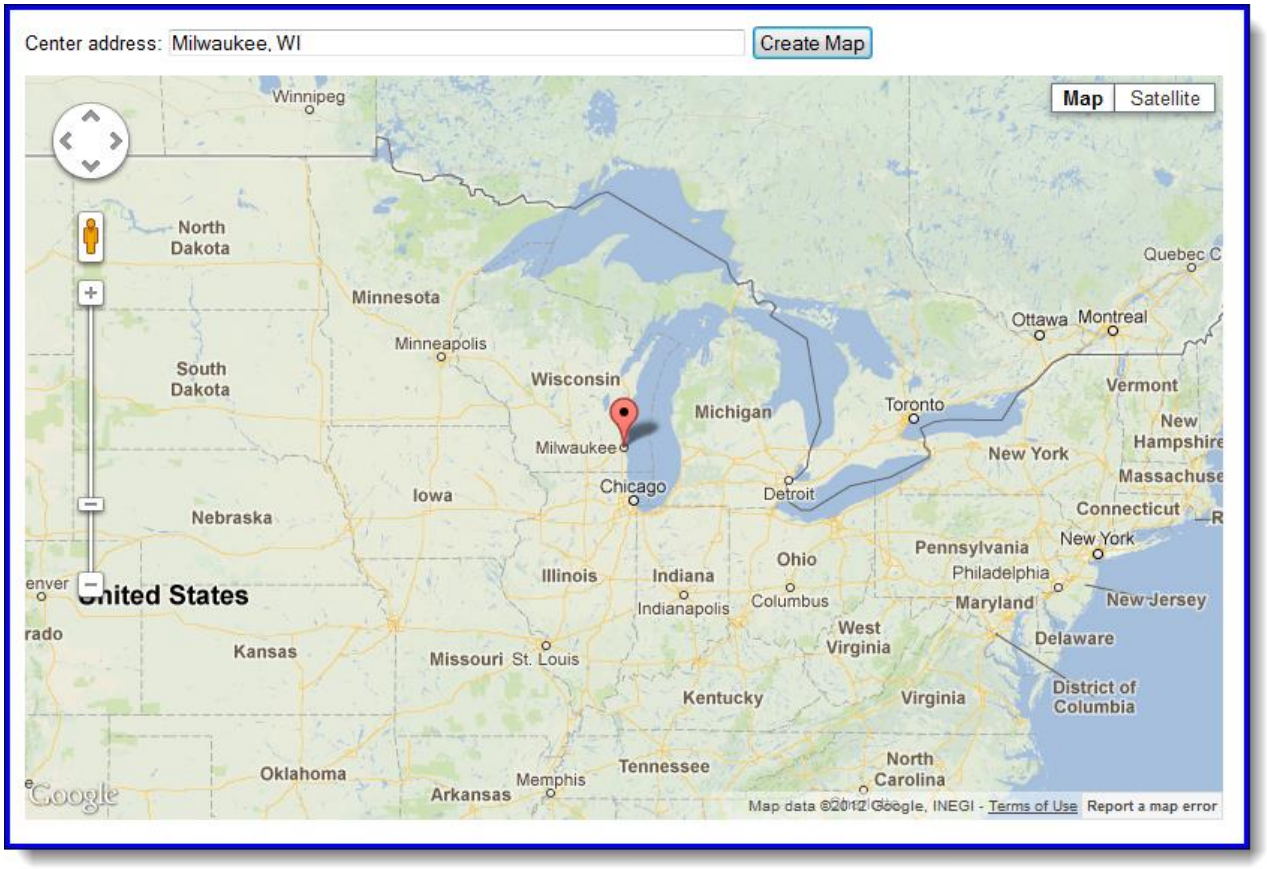
The jQuery (continued)

```
        // get directions
        directionsService.route(request,
           function(result, status) {
              if
                 (status == google.maps.DirectionsStatus.OK)
              {
                 directionsRenderer.setDirections(result);
              }
        });
    }
});
```

Exercise 16-1: Enhance a map application

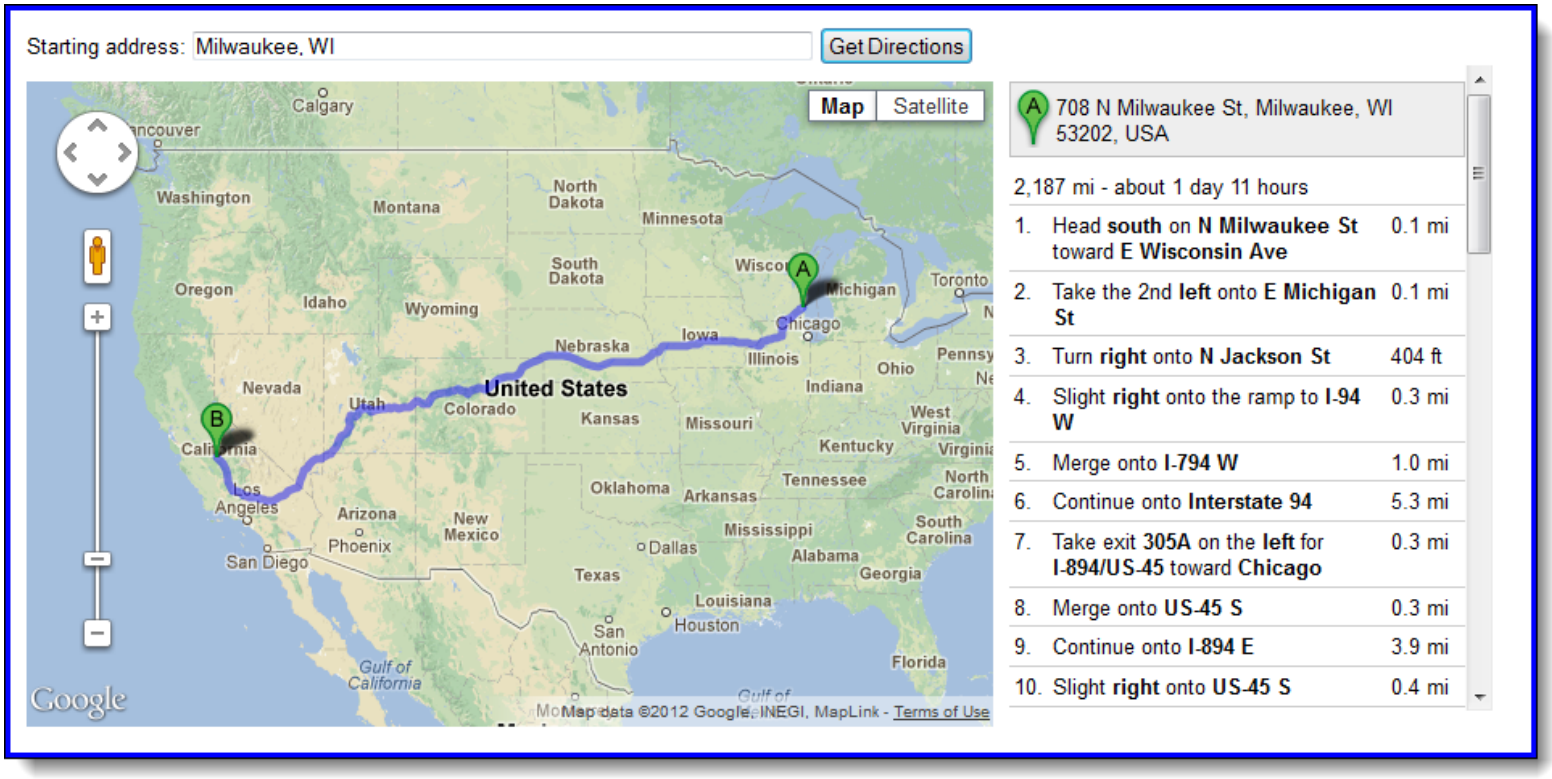


Extra 16-1: Create a Google map



The center of the map should be at the address that the user enters.

Extra 16-2: Provide Google Map directions



The starting address should be the one the user enters.

The destination address should be the one used for the center of the map.

Short 16-1: Add markers to a map

Estimated time: 15 to 20 minutes.