Extracted from:

Web Development Recipes

The Pragmatic Bookshelf

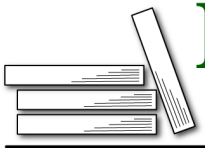
Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Web Development Recipes

Brian P. Hogan,
Chris Warren,
Mike Weber,
Chris Johnson,
and Aaron Godin

edited by Susannah Davidson Pfalzer



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Susannah Pfalzer (editor)
Potomac Indexing, LLC (indexer)
Kim Wimpsett (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-93435-683-8
Printed on acid-free paper.
Book version: P1.0—January 2012

Problem

Users want simple methods to locate their destination, and they want that information quickly in an easy and accessible manner. While addresses and written directions work, the simplest way is to glance at a map, memorize the street number, and grab your keys and go. By including a map on our site, we immediately give users a sense of where things are located and how they can get there.

Ingredients

- The Google Maps API

Solution

Using the Google Maps API, we can bring the power and functionality of Google Maps into our own application. We can render maps of two types: static and interactive. The static map is an image that we can insert into our page, whereas the interactive map allows for zooming and panning. The Google Maps API supports any programming language that can make a request to Google's servers. The documentation includes a lot of JavaScript examples, which is perfect for our needs.[1]

We can use the API to accomplish any task that a user could accomplish in the full application. We can render maps of two types: static and interactive. The static map is an image that we can insert into our page, whereas the interactive map allows for zooming and panning.

Along with rendering maps, the JavaScript API lets us insert other elements on the maps. We can place markers and bind mouse events to the markers. We can also create pop-out dialogs that show information directly within the map. We can show street views, geolocate the user, create routes and directions, and draw custom models on the map. The sky is in fact the limit until Google launches its space program and takes over NASA.[2]

We're working with a local university to develop a map for their web page for new visitors. The Admissions office wants to show these visitors where they can find places to eat as well as where to park. We'll create an interactive map that contains markers and information by using the JavaScript Google Maps API.

1. http://code.google.com/apis/maps/documentation/javascript/reference.html
2. http://www.google.com/space

Let's start off by creating a basic HTML document. We will declare the <DOCTYPE> as HTML5 as a recommendation from Google; however, if you can't use <DOCTYPE html> in your application, you're not explicitly required to do so.

Download googlemaps/map_example.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Freshman Landing Page</title>
    <style>
    </style>
    <script type="text/javascript">
    </script>
  </head>
  <body>
  </body>
</html>
```

Next, we'll include the Google Maps JavaScript API in our document. To make this request, we need to define whether our application is using a sensor to determine our user's location. Since this is not within the scope of the tutorial, we will set it to false.

Download googlemaps/map_example.html

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

The API requires a to act as a container for the map, so we'll add that to our page.

Download googlemaps/map_example.html

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

The map will scale to the size of this container, so let's set dimensions on this with CSS, by adding it to a new <style> section in our page's <head> region like this:

Download googlemaps/map_example.html

```
#map_canvas {
  width: 600px;
  height: 400px;
}
```

This container is now ready to hold a map that is 600x400 pixels. Let's go fetch some data.

Loading the Map with JavaScript

At the bottom of our <head> region, let's add a <script> block to hold the code that will initialize our map. We'll create a function called loadMap() to load the map based on our latitude and longitude, and we'll make this happen when the browser window loads. If you're using a framework such as jQuery in your project, you can do the loading of the map inside of your DOM-ready call, but we'll do this with vanilla JavaScript for our example.

Download googlemaps/map_example.html

```
window.onload = loadMap;
```

Next, we'll create the loadMap() function. Since we're not using a sensor, we'll hard-code our latitude and longitude. These coordinates define the center point of the map. To find these values, we have a few options. We could navigate to Google Maps, find what we want to center our map on, right-click a pin, and select "What's here?" The values for latitude and longitude appear in the search box. Alternatively, we can use Google Maps Lat/Long Popup.[3] This application allows us to click a location to find our values.

Download googlemaps/map_example.html

```
function loadMap() {
  var latLong = new google.maps.LatLng(44.798609, -91.504912);

  var mapOptions = {
    zoom: 15,
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    center: latLong
  };

  var map = new google.maps.Map(document.getElementById("map_canvas"),
        mapOptions);
}
```

Within this function, we create an object to hold some options for our map. We can define the type of map we want, a zoom value, and more. The zoom requires some experimentation; the higher the number, the further in it zooms. A value of 15 works well for street-level maps.

We can change how the map appears by setting a different mapTypeId. Note that zoom values along with maximum ranges for zoom change when changing map type. You can find a reference for map types in the Google Maps API documentation.[4]

3. http://www.gorissen.info/Pierre/maps/googleMapLocationv3.php
4. http://code.google.com/apis/maps/documentation/javascript/reference.html#MapTypeId

Finally, we create the map. The Map constructor requires that we pass the DOM element that will hold the map along with our object containing the options. When we load this page in our browser, as shown in Figure 21, *The initial map*, on page 9, we see a map centered on our desired location.

Creating Marker Points

To show incoming freshman where they can go to get a bite to eat or otherwise be social, we will create markers on the map. A marker in Google Maps is one of many overlays that we can add. Overlays respond to a click event, and we will use this to show an info window when the marker is clicked.

Since we already have our map, creating the marker is as simple as invoking the constructor and passing some options.

Download googlemaps/map_example.html

```
mogiesLatLong = new google.maps.LatLng(44.802293, -91.509376);
var marker = new google.maps.Marker({
  position: mogiesLatLong,
  map: map,
  title: "Mogies Pub"
});
```

To define a marker, we pass the latitude and longitude coordinates, the map that will hold the marker, and a title that appears when we hover over the marker.

Next, let's create the info window that appears when this marker is clicked. To create an info window, invoke the constructor.

Download googlemaps/map_example.html

```
var mogiesDescription = "<h4>Mogies Pub</h4>" +
  "<p>Excellent local restaurant with top of the line burgers and sandwiches.</p>";
var infoPopup = new google.maps.InfoWindow({
  content: mogiesDescription
});
```

Finally, we need to add an event handler to the marker. Using the Google Maps event object, we add a listener to open the info window.

Download googlemaps/map_example.html

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

When we click the marker, a new window shows up that gives us information about the location, as you can see in Figure 22, *A clicked marker*, on page 10.

Figure 21—The initial map

We can add any amount of HTML that we wish to the window. This gives us the freedom to show large amounts of information. From here, we can gather the coordinates of other points of interest and build the rest of the map.

Further Exploration

We have only scratched the surface of what can be accomplished with the Google Maps API. Along with markers, there are other layers of interaction that make the map more usable for your customers. You can create directions, map routes, use geolocation, and even add street views. Each of these features is well explained in the Google Maps API documentation,[5] and there are a number of working examples to follow along with.

Google Maps is just one component of the Google APIs. To see a full list of Google APIs, take a look at the Google APIs and Developer Products Page.[6]

Also See

- Recipe 17, *Building a Simple Contact Form*, on page ?

5. http://code.google.com/apis/maps/documentation/javascript/reference.html
6. http://code.google.com/more/table

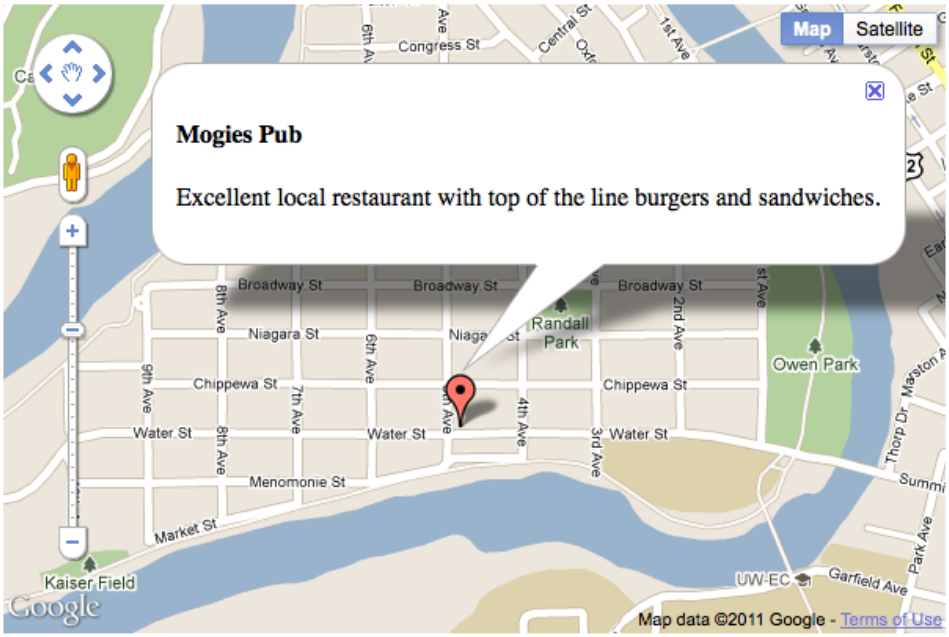


Figure 22—A clicked marker

- Recipe 18, *Accessing Cross-site Data with JSONP*, on page ?
- Recipe 19, *Creating a Widget to Embed on Other Sites*, on page ?