

Table of Contents

Chapter 2. Introducing the Google Maps API.....	1
2.1. Hacks 10–16: Introduction.....	1
Hack 10. Add a Google Map to Your Web Site.....	2
Hack 11. Where Did the User Click?.....	8
Hack 12. How Far Is That? Go Beyond Driving Directions.....	11
Hack 13. Create a Route with a Click (or Two).....	17
Hack 14. Create Custom Map Markers.....	20
Hack 15. Map a Slideshow of Your Travels.....	27
Hack 16. How Big Is the World?.....	30

Chapter 2. Introducing the Google Maps API

2.1. Hacks 10–16: Introduction

The Google Maps site at <http://maps.google.com> is awesome, with an easy user interface, one-box searching, and integrated satellite imagery. But it gets better! The Google Maps team has made it possible to include Google Maps with almost all of its great features onto your own web pages. They have done this by providing an open Application Program Interface, or API.

An API defines a standard way for one program to call code that lives within another application or library. The Google API defines a set of JavaScript objects and methods that you use to put maps on your own web pages.

Before Google Maps, it was much harder to put simple maps on your own pages. MapQuest had a program that let you create a link to a map. You could not embed the map on your own page, you could not put it into a frame, you couldn't even use `Target=_new` to open a new browser window.

There were—and are—open source solutions to generate maps. For example, the UMN Mapserver (<http://mapserver.gis.umn.edu>) is very powerful, allowing you to do things that Google Maps cannot yet equal, but there is a rather steep learning cliff. There are also industry standards for web mapping promulgated by the Open Geospatial Consortium (OGC) at <http://www.opengeospatial.org>.

The OGC-defined Web Mapping Service (WMS) and Web Feature Service (WFS) standards define a powerful web services interface to geospatial data. There are now free and easy-to-use open source clients for WMS and WFS data. For example, you can use the open source JavaScript library from <http://openlayers.org/> to access public data sources and put free maps on your page.

These are great options that either were not available or were not easy to use when the Google Maps API was released. Some folks argue that Google did nothing new. There was free data out there, and there were web-enabled GIS systems. But clearly Google did something.

Creating feature-complete interfaces to geographic data that are so complex as to be impenetrable by all but the highest of high priests seems to be nearly inevitable. Programmers start with a simple model, and then the world, the data, and the weight of history intrude to "complexify" our models and break our metaphors. When Walt Kelly wrote "We have met the enemy and he is us," he could have been describing the creators of most geospatial apps.

Somehow Google avoided that trap. As GPSBabel author Robert Lipe says, "We've seen an explosion in applications using the API. It's easy enough that you don't really need to be either a geo-geek, a cartographer, or a programming jock to produce some really usable results."

Google has provided a readable introduction to the API, available at <http://www.google.com/apis/maps/>. They have created a "Hello World" of Google Maps that embeds a Google Map on a web page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/
  TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Google Maps JavaScript API Example - simple</title>
    <script src="http://maps.google.com/maps?file=api&v=1&key=abcdefg"
type="text/javascript"></script>
  </head>
  <body>
```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

```

<div id="map" style="width: 300px; height: 300px"></div>
<script type="text/javascript">
//

    if (GBrowserIsCompatible()) {
        var map = new GMap(document.getElementById("map"));
        map.centerAndZoom(new GPoint(-122.141944, 37.441944), 4);
    }
//]]&gt;
&lt;/script&gt;
&lt;/body&gt;

&lt;/html&gt;
</pre>
</div>
<div data-bbox="112 268 882 309" data-label="Text">
<p>You need a bit of glue in the form of the above <code>div</code> element and the <code>script src=</code> line, but there are only two important lines of code required to create and display a map centered on a specific point. The key code in this example creates a new <code>GMap</code> object and then centers the map on a specific point:</p>
</div>
<div data-bbox="144 330 613 356" data-label="Text">
<pre>
var map = new GMap(document.getElementById("map"));
map.centerAndZoom(new GPoint(-122.141944, 37.441944), 4);
</pre>
</div>
<div data-bbox="112 370 889 398" data-label="Text">
<p>In programming terms, the first line creates a new variable that contains a new <code>GMap</code> object. The <code>GMap</code> object will be linked with the section of the page marked with <code>&lt;div id="map"&gt;</code>. The second line then calls the <code>centerAndZoom</code> method of the <code>GMap</code> class.</p>
</div>
<div data-bbox="112 414 878 429" data-label="Text">
<p>The <code>GMap</code> class represents a single map on your page (yes, you can have multiple maps on the same page, each displaying a unique view).</p>
</div>
<div data-bbox="112 445 888 485" data-label="Text">
<p>There are other classes that provide additional features, such as <code>GPoint</code> to define a point and <code>GPolyLine</code> to define a line (there is not yet true polygon support), as well as a built-in XML parser and XSLT processor. Take a look at the API documentation at <a href="http://www.google.com/apis/maps/documentation">http://www.google.com/apis/maps/documentation</a>.</p>
</div>
<div data-bbox="112 526 537 545" data-label="Section-Header">
<h2>Hack 10. Add a Google Map to Your Web Site</h2>
</div>
<div data-bbox="116 579 184 615" data-label="Image">
<img alt="Hack #10 icon: a red thermometer with the word 'HACK' and the number '10' inside it."/>
</div>
<div data-bbox="112 633 430 647" data-label="Section-Header">
<h3>Here's how to get started using the Google Maps API.</h3>
</div>
<div data-bbox="112 663 889 701" data-label="Text">
<p>At O'Reilly's Where 2.0 conference on June 29, 2005, Google announced an official and documented API for Google Maps. The API makes it possible for anyone to add a Google Map to a web page by cutting and pasting a few lines of JavaScript from the Google Maps Developer's site.</p>
</div>
<div data-bbox="112 718 888 783" data-label="Text">
<p>People reacted to the new API in one or more ways. My first act was to scratch my own itch by writing a bit of code to display my GPS waypoints on a Google Map. Fortunately, better GPX-to-Google Maps solutions have been created, one of which is documented in "View Your GPS Tracklogs in Google Maps" [<a href="#">Hack #37</a>]. After scratching that itch, I looked to our Geocoder.us site. Schuyler had spent a lot of time figuring out the Census Bureau's public TIGER/Line Map Server API, and how to display the resulting map with a neat little zoomable interface. The results were slow and clunky, but they worked.</p>
</div>
<div data-bbox="112 798 889 838" data-label="Text">
<p>The Google Maps API gets rid of the need for that level of head scratching! The march of progress in computers (possibly in society at large) works by first figuring out ways to do new things, and then progressively making those tasks easier, and leaving the old practitioners to eat cat food and write programs for their Osborne luggable computer.</p>
</div>
<div data-bbox="112 883 408 898" data-label="Section-Header">
<hr/>
<h2>Chapter 2. Introducing the Google Maps API</h2>
</div>
<div data-bbox="112 896 340 908" data-label="Text">
<p>Google Maps Hacks By Schuyler Erle, Rich Gibson</p>
</div>
<div data-bbox="112 905 425 917" data-label="Text">
<p>ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006</p>
</div>
<div data-bbox="112 914 889 944" data-label="Text">
<p>No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.</p>
</div>
<div data-bbox="517 941 888 963" data-label="Text">
<p>Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147<br/>Copyright 2006, Safari Books Online, LLC.</p>
</div>
```

I used Google Maps to bring the geocoder.us site into the protective embrace of the Google Maps API. Geocoder.us, online at <http://geocoder.us/>, is a free U.S. address geocoder. You can go to the web site and get the latitude and longitude for a U.S. street address. You can also use a web service interface to get the latitude and longitude automatically for a group of addresses [Hack #62]. You can geocode using Google Maps by scraping their search results, but it's not a part of the official API, and doing so violates Google's terms and conditions of service. By contrast, the Geocoder.us site is based on free data without limited terms of service for non-commercial use.

Figure 2-1 shows the results of geocoding the address of O'Reilly Media's headquarters with the original TIGER/Line map, with a pushpin showing the location of the address that we just looked up. We'd like to replace this somewhat slow map generated by the Census Bureau with the much faster, more attractive, and more easily navigable maps offered by Google Maps. (The original Geocoder.us map view can be seen at http://geocoder.us/demo_tiger.cgi.)

Figure 2-1. The Census Bureau map originally used by <http://geocoder.us/>

geocoder.us
find the latitude & longitude of any US address - for free

Address	1005 Gravenstein Hwy N Sebastopol CA 95472
Latitude	38.411908 °
Longitude	-122.842232 °

Search for another address:

2.2.1. Get a Developer Key

The first step in putting a Google Map on your page is to generate a developer's key, which is an alphanumeric string that identifies your web site to Google, and helps them track usage of Google Maps. Having to sign up for a developer's key can be something of an annoyance, but it's a small price to pay for being able to include free (as in beer) maps on your web site with such relative ease.

You'll need a distinct developer's key for each directory on your site that includes Google Maps. You don't need a key for each individual web page or script. So if you have several pages that generate calls to Google Maps from the same directory, you only need one key.

Fortunately Google has made getting developer's keys as easy as filling in a web form. The Google Maps API page is at <http://www.google.com/apis/maps/>. This includes links to documentation, examples, Terms of Use, and the page to get your key. There is a human version of the Terms of Use, then the full legalese version. Figure 2-2 shows the form with the URL we want to use for our maps. You must agree to the Terms of Service, then click Generate API Key.

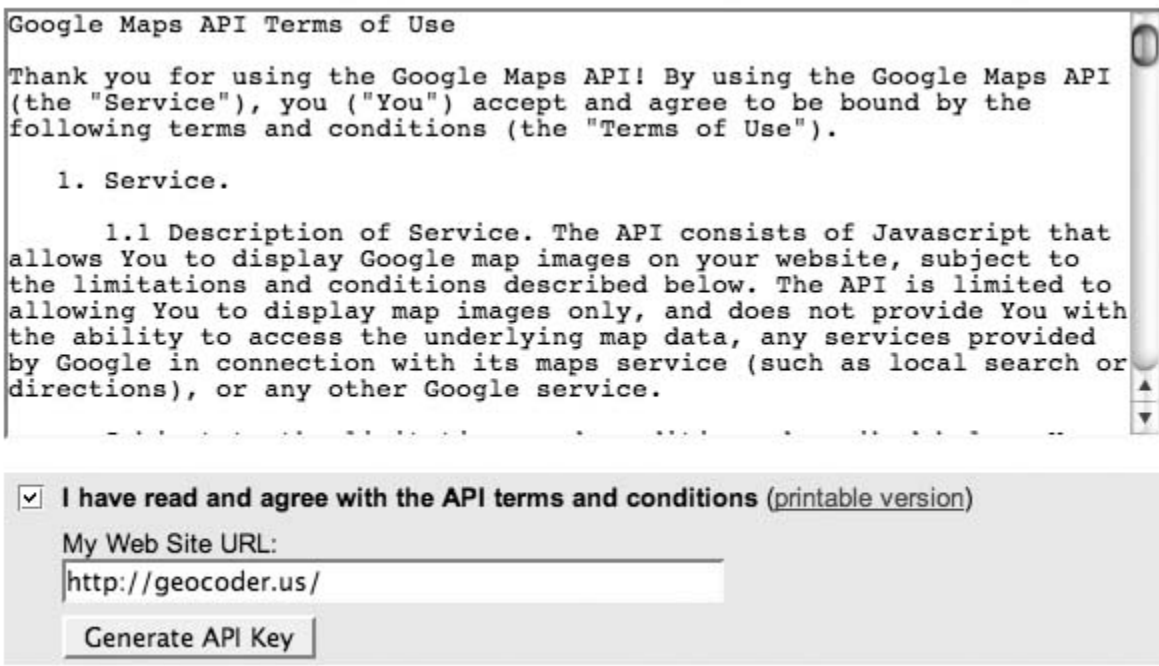
Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-2. Enter a server and path to generate a developer's key



Google Maps API Terms of Use

Thank you for using the Google Maps API! By using the Google Maps API (the "Service"), you ("You") accept and agree to be bound by the following terms and conditions (the "Terms of Use").

1. Service.

1.1 Description of Service. The API consists of Javascript that allows You to display Google map images on your website, subject to the limitations and conditions described below. The API is limited to allowing You to display map images only, and does not provide You with the ability to access the underlying map data, any services provided by Google in connection with its maps service (such as local search or directions), or any other Google service.

☒ I have read and agree with the API terms and conditions ([printable version](#))

My Web Site URL:



In our case, we wanted to enable Google Maps for a single script on our server. If you want to enable Google Maps for a whole directory, you can leave off the script name and just specify the host name and directory portion of the URL. Unfortunately, the API key isn't good for directories inside the one you specify, just the files and scripts in that directory.

Almost instantly, a key will be generated, along with an example web page that Google refers to this as the "Hello World" of Google Maps. To put this on your web site, copy the HTML/JavaScript section in [Example 2-1](#) and paste it into a new file on your own web site in the directory that you used when you created the developer's key.

Example 2-1. Google Maps "Hello World"

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script
      src="http://maps.google.com/maps?file=api&v=1&key=[your API key]"
      type="text/javascript"></script>
  </head>
  <body>
```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
 Copyright 2006, Safari Books Online, LLC.

```

<div id="map"
    style="width: 500px; height: 400px; border: 1px solid #979797"></div>
<script type="text/javascript">
//

var map = new GMap(document.getElementById("map"));
map.addControl(new GSmallMapControl());
map.centerAndZoom(new GPoint(-122.1419, 37.4419), 4);

//]]&gt;
&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>
</div>
<div data-bbox="149 308 211 361" data-label="Image">
<img alt="A small yellow square icon with a black and white panda face inside, surrounded by small black dots."/>
</div>
<div data-bbox="221 306 809 358" data-label="Text">
<p>Developer keys work only when they are used on a web page that lives in the server and directory that you specified when you created the key. So you can't copy this listing and have it work until you change the developer's key to match your own. In general, most of the code examples in this book will require you to substitute your own valid developer key in order for them to work.</p>
</div>
<div data-bbox="112 424 264 440" data-label="Section-Header">
<h2>2.2.2. Hello, World!</h2>
</div>
<div data-bbox="112 489 878 516" data-label="Text">
<p>The "Hello World" page shown in <a href="#">Example 2-1</a> is a standard HTML page, with a bit of JavaScript. The first interesting part is the opening script element:</p>
</div>
<div data-bbox="145 539 679 576" data-label="Text">
<pre>
&lt;script
src="http://maps.google.com/maps?file=api&amp;v=1&amp;key=[Your API Key]"
type="text/javascript"&gt;&lt;/script&gt;
</pre>
</div>
<div data-bbox="112 591 889 630" data-label="Text">
<p>This imports the Google Maps JavaScript library into our page. A JavaScript compliant browser will automatically fetch the contents of the provided URL. Google can then compare the developer's key and the server name and path that is included in the HTTP headers of your request with their records, to see if they match.</p>
</div>
<div data-bbox="149 669 211 722" data-label="Image">
<img alt="A small yellow square icon with a black and white panda face inside, surrounded by small black dots."/>
</div>
<div data-bbox="221 667 825 720" data-label="Text">
<p>The <code>v=1</code> parameter in the above URL is important, because it specifies the Google Maps API version that your script expects. If Google ever changes its API in such a way that backwards compatibility is broken, the <code>v</code> parameter will allow your script to continue to function with the original API and give you some breathing room to update your code to the newer version of the API.</p>
</div>
<div data-bbox="112 762 310 776" data-label="Text">
<p>The next three interesting lines are:</p>
</div>
<div data-bbox="145 797 580 835" data-label="Text">
<pre>
var map = new GMap(document.getElementById("map"));
map.addControl(new GSmallMapControl());
map.centerAndZoom(new GPoint(-122.1419, 37.4419), 4);
</pre>
</div>
<div data-bbox="112 883 408 899" data-label="Section-Header">
<h2>Chapter 2. Introducing the Google Maps API</h2>
</div>
<div data-bbox="112 896 340 908" data-label="Text">
<p>Google Maps Hacks By Schuyler Erle, Rich Gibson</p>
</div>
<div data-bbox="112 905 425 917" data-label="Text">
<p>ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006</p>
</div>
<div data-bbox="112 914 889 944" data-label="Text">
<p>No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.</p>
</div>
<div data-bbox="517 941 888 963" data-label="Text">
<p>Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147<br/>Copyright 2006, Safari Books Online, LLC.</p>
</div>
```

These lines are pretty much self explanatory (for an object-oriented JavaScript programmer). But you don't need to understand much to put powerful maps on your own pages!

By default the size of the map is determined by the size of the HTML element that contains the map. In this example, we are using the `div` element to define a division in the page, which provides an area that you can control and format independently from other parts of the page.

The first line creates a new `GMap` object and places it within the `div` named `map`. (There's nothing magic about the name of the `div` element, by the way—we could call it "Tim," and so long as the JavaScript mentioned the same name, it would still work.) The next line adds the small pan and zoom control to the map, and the third line centers and zooms the map to longitude -122.1419, latitude 37.4419 at zoom level 4.

In our example, the `div` element is 500 x 400 pixels high and has a 1-pixel-wide gray border around the edge. You can also specify the width and height in percentages, such as `style="width: 50%; height: 40%"`. The border itself is totally optional, but it does set the map off nicely from the rest of the page.

```
<div id="map"
  style="width: 500px; height: 400px; border: 1px solid #979797"></div>
```

The *demo.cgi* page at <http://geocoder.us/> was already template driven, so to add Google Maps functionality I added the `script=` line to load the Google Maps library, and then included these lines in my template:

```
<div id="map" style="width: 500px; height: 300px; border: 1px solid
#979797"></div>
<script type="text/javascript">
//

var map = new GMap(document.getElementById("map"));
map.addControl(new GSmallMapControl());
map.centerAndZoom(new GPoint([% long %], [% lat %]), 4);

var point = new GPoint([% long %], [% lat %]);
var marker = new GMarker(point);
map.addOverlay(marker);

//]]&gt;
&lt;/script&gt;</pre>
</div>
<div data-bbox="112 587 888 640" data-label="Text">
<p>The map will automatically size itself to fit within the <code>&lt;div id="map"..."&gt;</code> tag. In our templating system (Perl's Template Toolkit, as it happens), <code>[% long %]</code> will be replaced with the contents of the variable <code>long</code>, or the longitude. The only differences from the sample code are that the sample constants for <code>lat</code> and <code>long</code> are replaced with variables that will be set in our program, and that a point marker is added for the location of the address the user looked up.</p>
</div>
<div data-bbox="112 682 393 698" data-label="Section-Header">
<h3>2.2.3. Getting Outside of Your Head</h3>
</div>
<div data-bbox="112 747 889 826" data-label="Text">
<p>The "Hello World" example presumes that the HTML <code>script</code> element that imports the Google Maps API library into your web page is nestled safe within the HTML document's <code>head</code> element. Certainly, this is the right place for it to go, but web browsers are perfectly capable of handling <code>script</code> elements elsewhere in an HTML document. Furthermore, situations will occur where you might want to include the API library from elsewhere—say, for example, one where you have an HTML templating system that provides a boilerplate header and footer for each page on your site. In this circumstance, you don't want the API library to be imported into every page on the site, because every page outside the directory associated with your developer's key will load up with a developer key error message.</p>
</div>
<div data-bbox="112 883 408 898" data-label="Section-Header">
<hr/>
<h2>Chapter 2. Introducing the Google Maps API</h2>
</div>
<div data-bbox="112 896 425 917" data-label="Text">
<p>Google Maps Hacks By Schuyler Erle, Rich Gibson<br/>ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006</p>
</div>
<div data-bbox="112 915 888 944" data-label="Text">
<p>No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.</p>
</div>
<div data-bbox="517 941 888 963" data-label="Text">
<p>Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147<br/>Copyright 2006, Safari Books Online, LLC.</p>
</div>
```

Fortunately, you can indeed import the API library almost anywhere in your document, so long as it appears before the JavaScript code that needs to use it. The only thing you really need to know is that some browsers—Internet Explorer, in particular—will wait for a script element to execute before rendering the rest of the page, to make sure that the JavaScript itself doesn't modify the page layout. For some reason, this behavior sometimes has a bad interaction with the Google Maps API when the script element is used outside the head—a JavaScript execution error is the most common result. The workaround is to add a `defer="defer"` attribute to the script element, which will tell the browser not to worry about it and get on with rendering the page. In that case, our earlier script element example looks like this:

```
<script src="http://maps.google.com/maps?file=api&v=1&key
    =[Your API Key]"type="text/javascript" defer="defer" ></script>
```

2.2.4. Getting Right to the Point

Once you've got a Google Map on your page, adding points to it is easy. You'll first create a new `GPoint` object, then create a marker icon at that point, and finally add that marker to the map. We'll look more at adding points and lines to Google Maps in other hacks. For now, enjoy [Figure 2-3](#), which shows a pretty Google Map replacing our TIGER map.

But is that (always) better? Are there reasons not to use Google Maps? Yes! Google Maps are great, and Google has a history and reputation of being the good guys, but it is a profit-making business and its goals might not be your goals. The Google Maps terms of service are extremely generous, but when you use Google Maps, you are relying on Google. There are restrictions on what you can do with Google Maps; for example, Google Maps cannot be used on a site that is inaccessible to the general public, such as a paid premium content site or a corporate intranet. There are limitations on volume, as well: if you expect more than 50,000 hits in a day, Google expects to hear from you first. You can't do certain things, such as scrape Google's images or remove its imprint from its imagery, and it has explicitly reserved the right to put ads on the maps at any time. You can read more about the fine details at <http://www.google.com/apis/maps/faq.html>, but you should also review the terms of use at <http://www.google.com/apis/maps/terms.html> to be on the safe side.

Figure 2-3. <http://geocoder.us/> with a Google Map

geocoder.us

find the latitude & longitude of any US address - for free

Address 1600 Pennsylvania Ave NW
Washington DC 20502


Latitude 38.898748 °

Longitude -77.037684 °

Search for another address:

1600 Pennsylvania Ave, Washington DC

Submit



= geocoder.us = © 2004-5 Locative Technologies = terms & conditions = contact us =

POWERED BY
PERL

CC
 SOME RIGHTS RESERVED

There are (at least currently) limits on the data available from Google. There is far more aerial and satellite data and map imagery available on the Web [[Hack #12](#)] from public Web Mapping Service (WMS) servers than is available from Google.

2.2.5. See Also

- Google Maps are free-as-in-beer but not free-as-in-speech. So if the power, beauty, and ease of use of Google Maps don't meet your needs, projects such as Geoserver (<http://geoserver.sf.net/>), Mapserver (<http://mapserver.gis.umn.edu/>), and the Ka-Maps client interface to Mapserver (<http://ka-maps.sf.net/>) may fill the bill. The downside, as is often the case with open source software, is that you may have to do more of the work yourself! O'Reilly's *Mapping Hacks* and *WebMapping Illustrated* have much more to say about free and open source mapping solutions.

Hack 11. Where Did the User Click?



Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Find the location of a click on a map and display it on your web page.

Google Maps makes it easy to put an interactive map on your web page. At <http://www.naffis.com/maphacks/latandlon.html> you can click on a map and have the corresponding latitude and longitude displayed in a Google Maps info box.

In [Figure 2-4](#) you can see the location of the Washington Monument.

Figure 2-4. The Washington Monument at 38.88941 N, 77.03517 W



This site solves the common problem of figuring out the coordinates of a location from a map view and is an example of the sort of quick hack that Google Maps has made possible. This page illustrates one way to get the latitude and longitude from a click on a map and display results in an info box. So how can you do it? At <http://mappinghacks.com/projects/gmaps/click.html> there is a simplified example of updating a form from the coordinates of a click on a map.

This is the Hello World map with three changes. An HTML form has been added to receive the latitude and longitude from the click event:

```
<form>
Latitude: <input type="text" value="38.4094" id="click_lat"
onclick="this.blur()">&nbsp;
Longitude: <input type="text" value="-122.8290" id="click_long"
onclick="this.blur()">&nbsp;
</form>
```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

In the script, the latitude and longitude used to set the initial map location with `centerAndZoom()` now comes from these form elements. This code defaults to starting at 38.4094 N, 122.8290 W. Change those values to change the initial focus of the map. The important change to the script is the addition of a `GEvent.addListener`. This code and the above form can be pasted into the body of your HTML page. Change the developer's key and you can capture clicks on a map:

```
<script src=
  "http://maps.google.com/maps?file=api&v=1&key=replacewithyourkey"
  type="text/javascript">
</script>

<div id="map" style="width: 400px; height: 300px"></div>
<script type="text/javascript">
//

    var map = new GMap(document.getElementById("map"));
    map.addControl(new GSmallMapControl());

    // center and zoom to the lat/long in the form
    map.centerAndZoom(new GPoint(
        document.getElementById('click_long').value,
        document.getElementById('click_lat').value), 3);

    GEvent.addListener(map, 'click',
        function(overlay, point) {
            if (point) {
                document.getElementById('click_lat').value = point.y;
                document.getElementById('click_long').value = point.x;
            }
        }
    );
//]]&gt;
&lt;/script&gt;</pre>
</div>
<div data-bbox="112 523 889 564" data-label="Text">
<p>This code adds a listener to the <code>GMap</code> object named <code>map</code>. If you click on the map, the code in the event handler will be run. The code is given both an overlay (a marker) and a point. If you click on a marker, the overlay will be set. If you don't click on a marker, then a <code>GPoint</code> object is given to the code.</p>
</div>
<div data-bbox="112 580 886 620" data-label="Text">
<p>These two lines are standard JavaScript. They ask the document for the value of the elements <code>click_lat</code> and <code>click_long</code>. The only elements with those names are the form elements. <code>point.x</code> and <code>point.y</code> are the longitude and latitude of the <code>GPoint</code> object that marks where you clicked.</p>
</div>
<div data-bbox="144 643 588 668" data-label="Text">
<pre>document.getElementById('click_lat').value = point.y;
document.getElementById('click_long').value = point.x;</pre>
</div>
<div data-bbox="112 683 326 696" data-label="Text">
<p>The end result is shown in <a href="#">Figure 2-5</a>.</p>
</div>
<div data-bbox="112 883 408 898" data-label="Section-Header">
<hr/>
<h2>Chapter 2. Introducing the Google Maps API</h2>
</div>
<div data-bbox="112 895 425 917" data-label="Text">
<p>Google Maps Hacks By Schuyler Erle, Rich Gibson<br/>ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006</p>
</div>
<div data-bbox="112 914 889 944" data-label="Text">
<p>No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.</p>
</div>
<div data-bbox="517 941 888 963" data-label="Text">
<p>Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147<br/>Copyright 2006, Safari Books Online, LLC.</p>
</div>
```

Figure 2-5. The click is back

Add the ability to capture the lat/long where the user clicks.

Latitude: Longitude:



The next steps are to add markers to the map from the click, populate a map from an external data source, and update an external data source based on the map.

Hack 12. How Far Is That? Go Beyond Driving Directions



Draw routes and calculate distances on your own Google Maps.

How far is it? That's a basic question we often ask of maps. Google Maps' driving directions answer that question, but driving directions are not (yet) accessible to the developer's API. More importantly, they simply give driving distances assuming the optimal route, where *optimal* is defined as getting there as quickly as possible in an automobile. They are not optimized for "scenic drive" or "safest bicycle route" or "quiet stroll" or "jog around the park."

There are at least two sites that allow you to create routes and calculate distances by clicking on maps. The Gmaps Pedometer at <http://www.sueandpaul.com/gmapPedometer/> shown in Figure 2-6 estimates cumulative distance—and even includes a calorie counter.

Use the standard map controls to zoom into your area of interest, and then click Start Recording. When you double-click a point on the map, it will recenter to that spot and add a marker there. The second time you click, the map will recenter to your new point, the marker will be moved, and a line will be drawn from the last point clicked to this one. Each time you do this, the Total Distance and Last Leg Distance fields will be updated.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-6. Sue and Paul's Gmaps Pedometer



Handling double clicks is a bit awkward and browser dependent. Sue and Paul are doing browser detection and then setting the appropriate event handler based on the browser:

```
if (navigator.appName == 'Microsoft Internet Explorer'){
    document.ondblclick = handleDblClick;
    bIsIE = true;
} else {
    window.ondblclick = handleDblClick;
    bIsIE = false;
}
```

Do you see the difference between the two `ondblclick` events? It is a difference in how they implement the Document Object Model. Internet Explorer handles double-clicks at the document level, hence `document.ondblclick` and everyone else (well, everyone else according to this code) uses the window object, so `window.ondblclick`. In both cases when there is a double-click the variable `bDoubleClickHappened` is set to true.

This will become important in a bit. You can't add a double-click listener with the Google Maps API, so the map does not directly capture the double-click event, but rather the `moveend` event, which according to the API documentation is "Triggered at the end of a discrete or continuous map movement. This event is triggered once at the end of a continuous pan."

This means that when there is a double-click event, Sue and Paul's `handleDblClick` function is called to set the `bDoubleClickHappened` variable. Next the Google Maps equivalent of `handleDblClick` is called.

Once Google Maps has finished the move or pan, the `moveend` event is triggered, the function set to listen for `moveend` events is called, and the anonymous function set in this code is called:

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

```

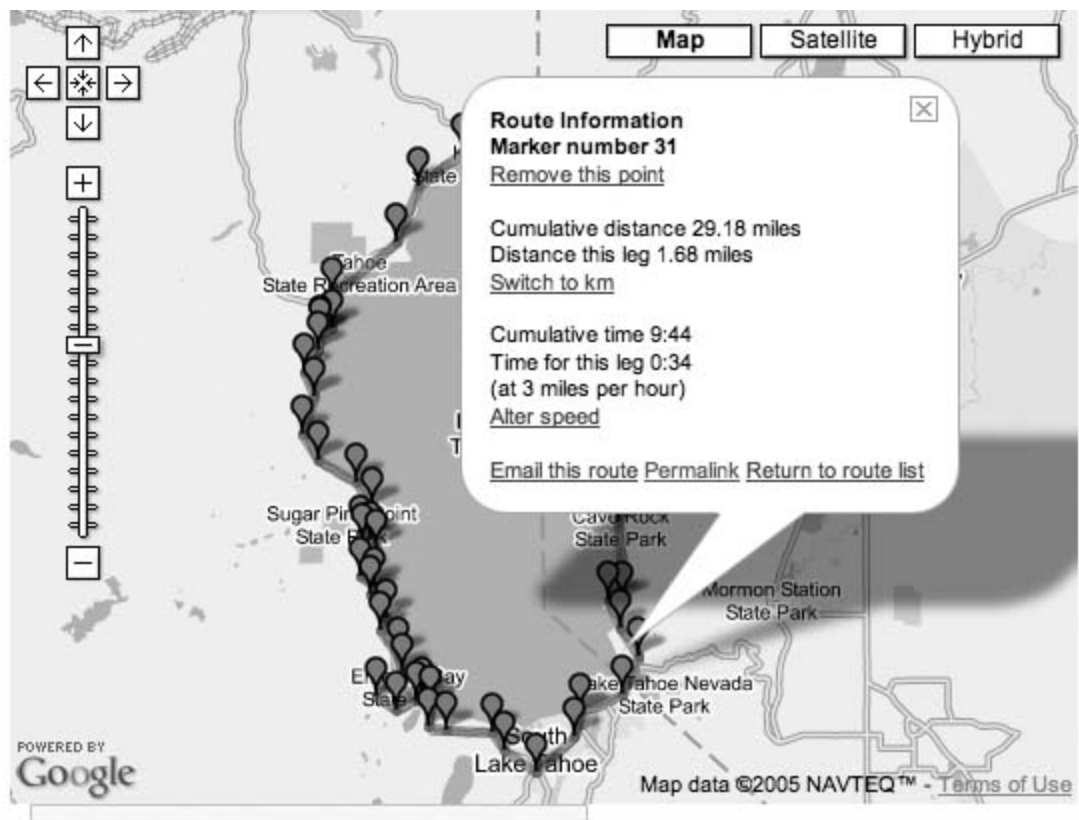
GEvent.addListener(map, "moveend", function() {
    if (bDoubleClickHappened){
        addLeg(map.getCenterLatLng().x, map.getCenterLatLng().y);
        drawPolyLine(gPointArray);
    }
    bDoubleClickHappened = false;
});

```

I love this code! It is an example of not always getting what you want, but finding a way to get what you need. We don't want to add a leg to our route on every move, just the ones that were initiated by a double-click. When the double-click handler was called, the `bDoubleClickHappened` variable was set. This code is called any time the map is moved, and if the map was double-clicked the `addLeg` and `drawPolyLine` functions are called.

Walk Jog Run, shown in [Figure 2-7](#), works in similar ways, but it captures single clicks. If you click on the map it asks if you want to start a new route. If you say yes, it captures each click, adds a marker, and draws a line connecting all of the points you've clicked.

Figure 2-7. Walk Jog Run



If this is the first click, the route will be empty, so `startRoute()` is called; otherwise, this is the continuation of a route, so this point is added to the list.

```

GEvent.addListener(map, 'click', function(overlay, point) {
    if (overlay) {
        /* do nothing */
    }
}

```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

```

    } else if (point) {
        if (route.length == 0) {
            startRoute(point);
        } else if (route[route.length-1].x != point.x ||
            route[route.length-1].y != point.y) {
            route.push(point);
            currentRouteId = null;
            drawRoute(route);
        }
    }
});

```

The prototype for a click event handler accepts an overlay and point parameter. The overlay parameter is set when the user clicks on an overlay (that is, a line or marker). Most of the time we handle click events on markers by setting a listener when the marker is created. The `drawRoute` method is called when a new point is added and then goes through the list of points in the route, setting up the text for the marker overlays and calling this `createMarker` code to draw them, and then drawing the polyline of the route.

```

function createMarker(point, mtext, icon) {
    var marker = new GMarker(point, icon);
    var html = "<b>Route Information</b><br />" + mtext;
    GEvent.addListener(marker, "click", function() {
        currentPoint = marker.point;
        map.centerAndZoom(currentPoint, map.getZoomLevel());
        marker.openInfoWindowHtml(html);
    });
    return marker;
}

```

This illustrates the creation of a marker with embedded text. If you want your markers to open, add a listener for the click event of the marker. The standard choice when clicking a marker is the `openInfoWindowHTML()` method for the marker. This pops up the standard HTML-enabled info window, but you can do anything when a marker is called.



In the wouldn't-it-be-cool-if category, wouldn't it be cool if there were an Internet-enabled game of Risk using Google Maps as the interface? You would have a map of the world with markers that represent the Risk countries. Clicking on the marker would bring up an HTML form with your options in regards to that country—from just getting information, to launching an attack. Given the pace of development on Google Maps, I suspect that googling for "Google Maps Risk" will bring up three different implementations by some time Tuesday (depending on which Tuesday).

Walk Jog Run aims to be the <http://del.icio.us> of maps for the running community. It lets you save your own routes as well as search and comment on the shared list of routes. You can look at the information relevant to any of the intermediate points on the route. Walk Jog Run shows you an estimated time for the total route and for each segment and allows you to delete any points from a route, unlike Sue and Paul's, which will only let you undo the last point.

Lines or Points? Both services let you add points by clicking. GMaps Pedometer shows a marker for your start and most recent position and hides the markers for the intermediate points. The result is a clean path overlaid on the map. Walk Jog Run leaves the markers on the map, which lets you view statistics for each segment of a completed route. They each have advantages.

Both GMaps Pedometer and Walk Jog Run aim to be full-featured sites. As a result, the code has a lot of detail that might make it hard to understand what is going on. Our page at <http://mappinghacks.com/projects/gmaps/lines.html> has another example of adding markers and lines, and then calculating distances in response to click events, as shown in [Figure 2-8](#).

Chapter 2. Introducing the Google Maps API

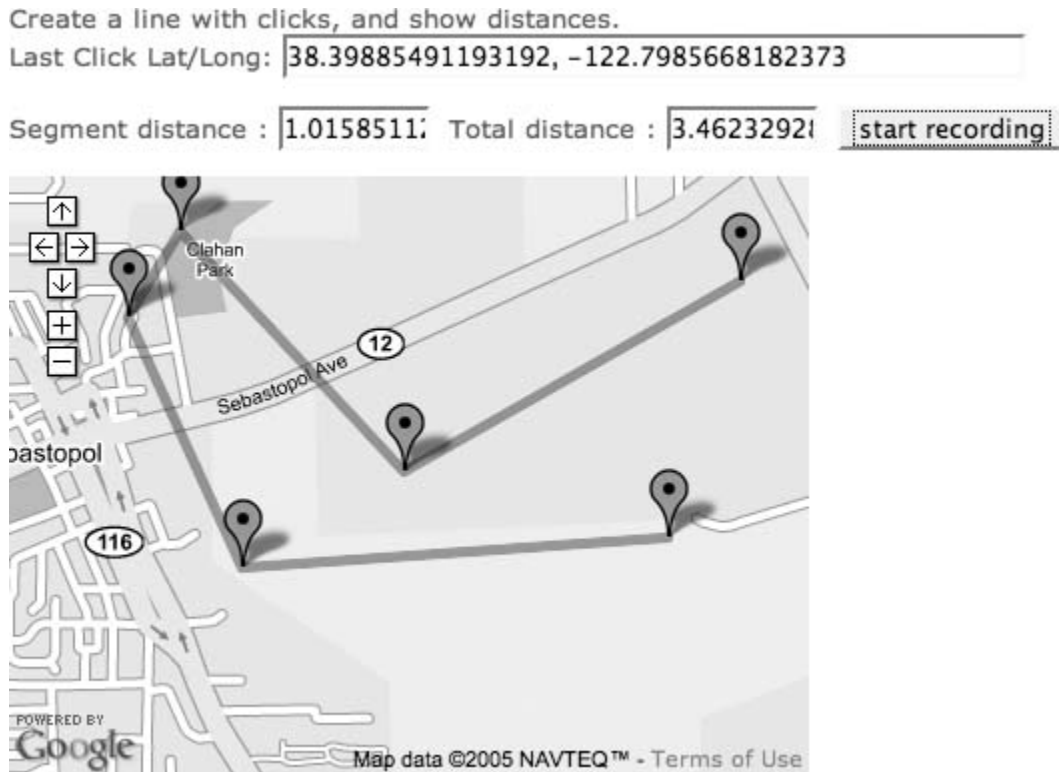
Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-8. Adding markers and lines



When the user clicks on the Start Recording button, the code sets up a listener to process click events and the `recording_flag` is set in the JavaScript. If it is set the current position is added to the arrays that hold our *x* and *y* positions (where, you'll recall *x* equals longitude, and *y* equals latitude), and the `drawRoute` function is called. Finally, the current latitude and longitude are shown in the form elements `click_lat` and `click_long`. Capturing clicks is described in more detail in "Where Did the User Click?" [[Hack #11](#)].

```

GEvent.addListener(map, 'click',
  function(overlay, point) {
    if (point) {
      if (recording_flag > 0) {
        addPoint(point.y, point.x, keepPoint);
        x_array.push(point.x);
        y_array.push(point.y);
        drawRoute();

        document.getElementById('click_lat').value = point.y;
        document.getElementById('click_long').value = point.x;
      }
    }
  }
); // end of GEvent.addListener

```

The `drawRoute()` function is a bit longer, but hopefully straightforward. The first trick when updating markers is to clear all the existing markers by calling `clearOverlays()`. Next the code walks the array of longitudes, `x_array`. Distances are calculated for the segment and the running distance of the route up to this point. The `segment_distance` and `total_distance` form elements are updated to show the distances.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

The point is then created and a marker added. The created point is added to the array points. Once all of the elements in the x and y arrays have been processed, the array of points is added as a new GPolyLine.

```
function drawRoute() {
  map.clearOverlays();
  var points = [];
  for (i = 0; i < x_array.length; i++) {
    if (i>0) {
      segment_distance_array[i] = calcDist(x_array[i-1], y_array[i-1],
        x_array[i], y_array[i]);
      total_distance_array[i] = total_distance_array[i-1] +
        segment_distance_array[i];
      document.getElementById('segment_distance').value =
        segment_distance_array[i];
      document.getElementById('total_distance').value =
        total_distance_array[i];
    } else {
      // initialize the first element distances to 0
      document.getElementById('segment_distance').value = 0;
      document.getElementById('total_distance').value = 0;
      total_distance_array[0] = 0;
      segment_distance_array[0] = 0;
    }
    var point = new GPoint(x_array[i], y_array[i]);
    points.push(point);
    var marker = new GMarker(point);

    // define the text that appears in the marker
    var html = "location <b>" + y_array[i] + ', ' + x_array[i] + "</b>";
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    map.addOverlay(marker);
  }
  map.addOverlay(new GPolyline(points));
}
```

This is not the only, or even best, way to do this! There is more than one way to do it!

Now, let's move on to calculating distances. Walk Jog Run and GMaps Pedometer use similar functions to calculate distance. I used the one from Walk Jog Run in my demo because it specifically had a Creative Commons Attribution-NonCommercial-ShareAlike license.

```
/* calcDist() function is from Adam Howitt
   Copyright Adam Howitt 2005
   Email: adamhowitt@gmail.com
   This work is licensed under a Creative Commons
   Attribution-NonCommercial-ShareAlike 2.5 License.
   http://creativecommons.org/licenses/by-nc-sa/2.5/
*/
function calcDist(lon1,lat1,lon2,lat2) {
  var r = 3963.0;
  var multiplier = 1;
  // var multiplier = currentUnit == "miles" ? 1 : MILESASKM;
  return multiplier * r * Math.acos(Math.sin(lat1/57.2958) *
    Math.sin(lat2/57.2958) + Math.cos(lat1/57.2958) *
    Math.cos(lat2/57.2958) * Math.cos(lon2/57.2958 -
    lon1/57.2958));
}:
```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Note that the variable `multiplier` has been commented out. In this code, I'm displaying the values only in miles. The multiplier represents the conversion factor from miles to whatever units you need. `MILESASKM` is *Miles as Kilometers*, the number of kilometers in one mile, or 1.609344. The multiplier is set using the ternary operator. If the current unit equals `miles` then the multiplier is 1 (as 1 mile equals 1 mile); otherwise, it is set to the number of kilometers in a mile. You don't need to understand this formula, but if you want to learn more, see "How to calculate distance in miles from latitude and longitude" at <http://www.meridianworlddata.com/Distance-Calculation.asp>.

The constant 3,963 is close enough to the radius of the earth in statute miles. 57.2958 is the number of statute miles (5,280 feet, as opposed to nautical miles) in one degree of latitude anywhere, or one degree of longitude at the equator. A nautical mile is defined as 1 minute of latitude (or 1 minute of longitude at the equator).

With the multiplier code commented out, you can copy this function into your own code and calculate distances between anything. Go distance crazy!

Hack 13. Create a Route with a Click (or Two)



You can even take Google Maps where the roads don't go.

Driving directions don't always take you where you want to go, or they may take you the wrong way. You can use a little Google Maps hack to build up your own list of points that can be saved in different formats that can be loaded into a GPS or other tool that supports the GPX standard.

The click-to-route tool is at <http://mappinghacks.com/projects/gmaps/clicktoroute.html>. You click on the map to create a continuous track. An example is shown in [Figure 2-9](#).

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-9. No roads—only walkers, horses, and bikes here!



Once you've created your route, click on one of the Export buttons. Clicking on CSV (Comma Separated Values) generates a pure-text file with the latitude and longitude separated with commas. GPX Track makes a GPX Tracklog file. GPX Route generates a set of points that can be loaded into a GPS. Here is a sample of the route as a CSV file:

```
38.4047068183193, -122.84743666648865
38.4041771393969, -122.84764051437378
38.403941725296505, -122.84796237945557
```

And here is an example of a GPX Tracklog:

```
<?xml version="1.0"?>
<gpx>
<trk><name>Google Maps Hacks is Good</name>
<trkseg>
<trkpt lat="38.41324840580697" lon="-122.84113883972168"></trkpt>
<trkpt lat="38.402688973080245" lon="-122.82877922058105"></trkpt>
<trkpt lat="38.4049085997449" lon="-122.84637451171875"></trkpt>
</trkseg>
</trk>
</gpx>
```



Many GPS units have a limit on the number of points that can be used in a route. This GPSBabel command will simplify your list of points to 30.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

```
gpsbabel -r -i gpx -f route.gpx \
-x simplify,count=30 -o gpx \
-F shorter_route.gpx
```

See "Load Driving Directions into Your GPS" [\[Hack #35\]](#) for more on reducing the number of points and loading a route file into your GPS.

2.5.1. The Code

This hack is almost identical to "How Far Is That? Go Beyond Driving Directions" [\[Hack #12\]](#), with the addition of one function that generates the formatted list of points, and buttons to call this function. The buttons are inserted with this HTML code:

```
<input type="button" value="Export CSV" onClick="exportPoints('csv'); ">
<input type="button" value="Export GPX Track"
onClick="exportPoints('track'); ">
<input type="button" value="Export GPX Route"
onClick="exportPoints('route'); ">
```

The `onClick` event is set to call the `exportPoints` function with a parameter to set the format of the exported points. This example shows the `exportPoints` function with the code for the GPX exports removed. The GPX format is simple XML, and that clutters up the example.

```
function exportPoints(format) {
    var export_string;
    if (format=='csv') {
        //csv header
        export_string = export_string + "latitude, longitude\n";
    }

    for (i = 0; i < x_array.length; i++) {
        var lon = x_array[i];
        var lat = y_array[i];
        if (format=='csv') {

            export_string = export_string + lat + ", " + lon + "\n";
        }
    }

    // write into document
    document.getElementById("output").value=export_string;
}
```

The results of your selected route will appear in a `textarea` below the map. You may need to scroll down to see the list. You can select the whole text area and paste it into your own document. Once you have a list of points to map back on Google Maps [\[Hack #37\]](#), export the points to your GPS [\[Hack #35\]](#), and even calculate driving directions between the points [\[Hack #36\]](#).

You can also use this technique to plan a trip or to explore more about a trip you took without a GPS (or a trip where the GPS didn't work because the darn satellite signals wouldn't penetrate the steel canyons of the city).

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Hack 14. Create Custom Map Markers



Adding custom markers to your Google Map can enhance its readability and appeal.

Almost immediately after the Google Maps API announcement, Jeff Warren made a hack that used custom icons to do a map depicting Star Wars ATATs attacking Google's home town, Palo Alto, as shown in [Figure 2-10](#). You can launch your own Imperial assault on Google's home base at <http://www.vestaldesign.com/maps/starwars.html>.

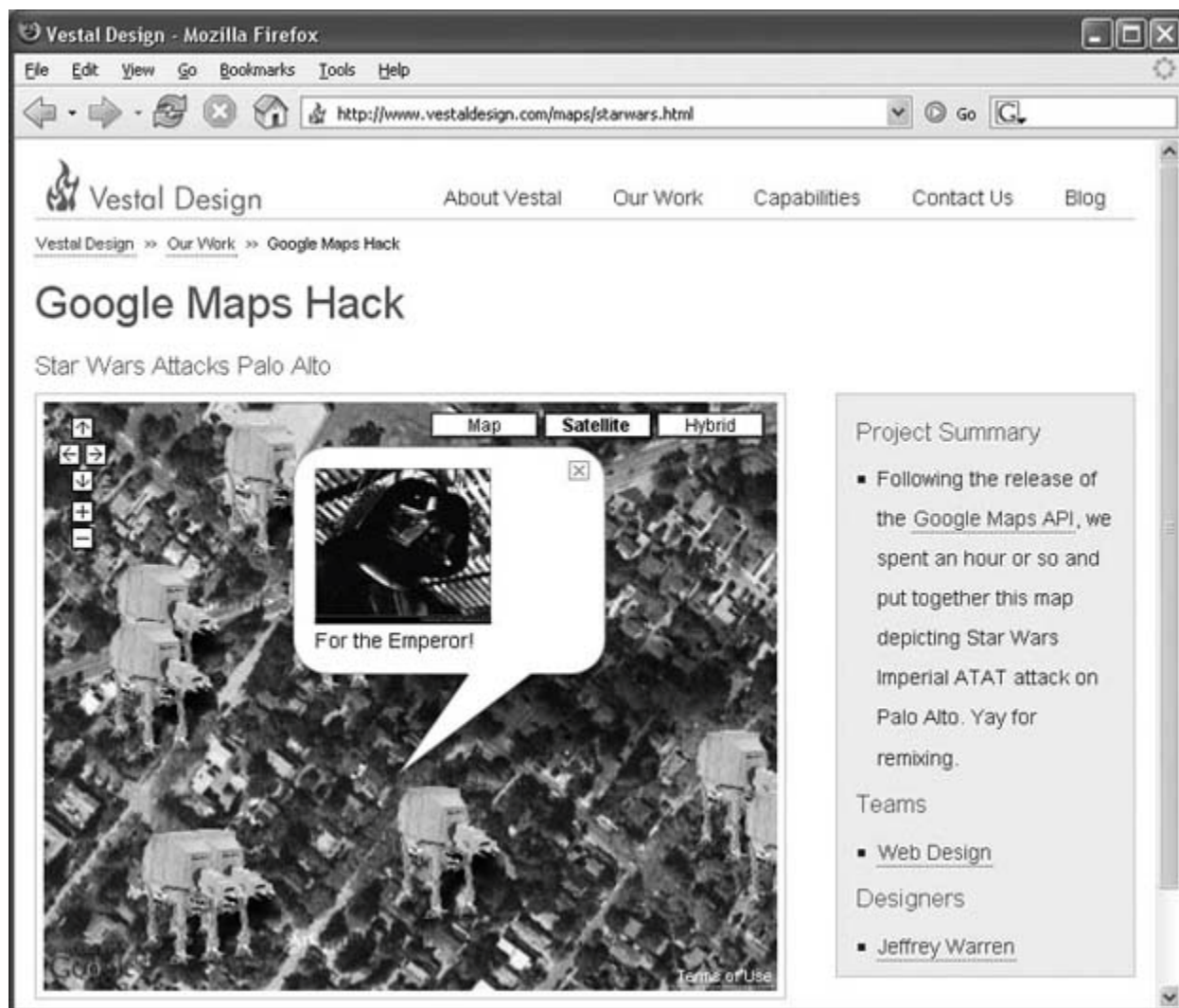
This hack immediately demonstrated the flexibility of Google's new API. If you wanted to use a house icon instead of the generic marker, you could. Likewise, if you wanted to make a multiplayer game using Google Maps, the API was flexible enough to allow you to let users submit their own icons.

To create an icon, you need two things: a foreground image for the icon and a shadow image in the PNG 24-bit file format. If you are only changing the color of the generic marker, you can reuse the generic shadow, but for this hack we're going to make something completely different.

2.6.1. Find the Right Foreground Image

Instead of doing something generic and boring, I decided to take a headshot of a friend of mine and turn it into a Google Maps marker icon! I grabbed a suitable shot from a digital photo, loaded it into Adobe Photoshop, and started erasing, as shown in [Figure 2-11](#). About halfway through, I could tell this was going to make a great foreground image.

Figure 2-10. You too can send Imperial ATATs to attack Google's headquarters



If you're not into commercial software, the GNU Image Manipulation Program, or GIMP, offers an open source alternative with the same basic features. You can find the GIMP at <http://www.gimp.org/>.

Once I finished erasing around Karl's head, I scaled down the image, cropped it, and expanded the canvas to give Karl's head some breathing room. I wouldn't want him to get claustrophobic, and when we start working on the shadow, we're going to need a bit of extra room. You can see the finished result in [Figure 2-12](#).

Finally, I went to the File → Save for Web option, and saved my file in the PNG-24 format with transparency. 24-bit PNG format is ideal for custom map icons, because it's lossless and the alpha layer support allows for some wonderful transparency effects.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

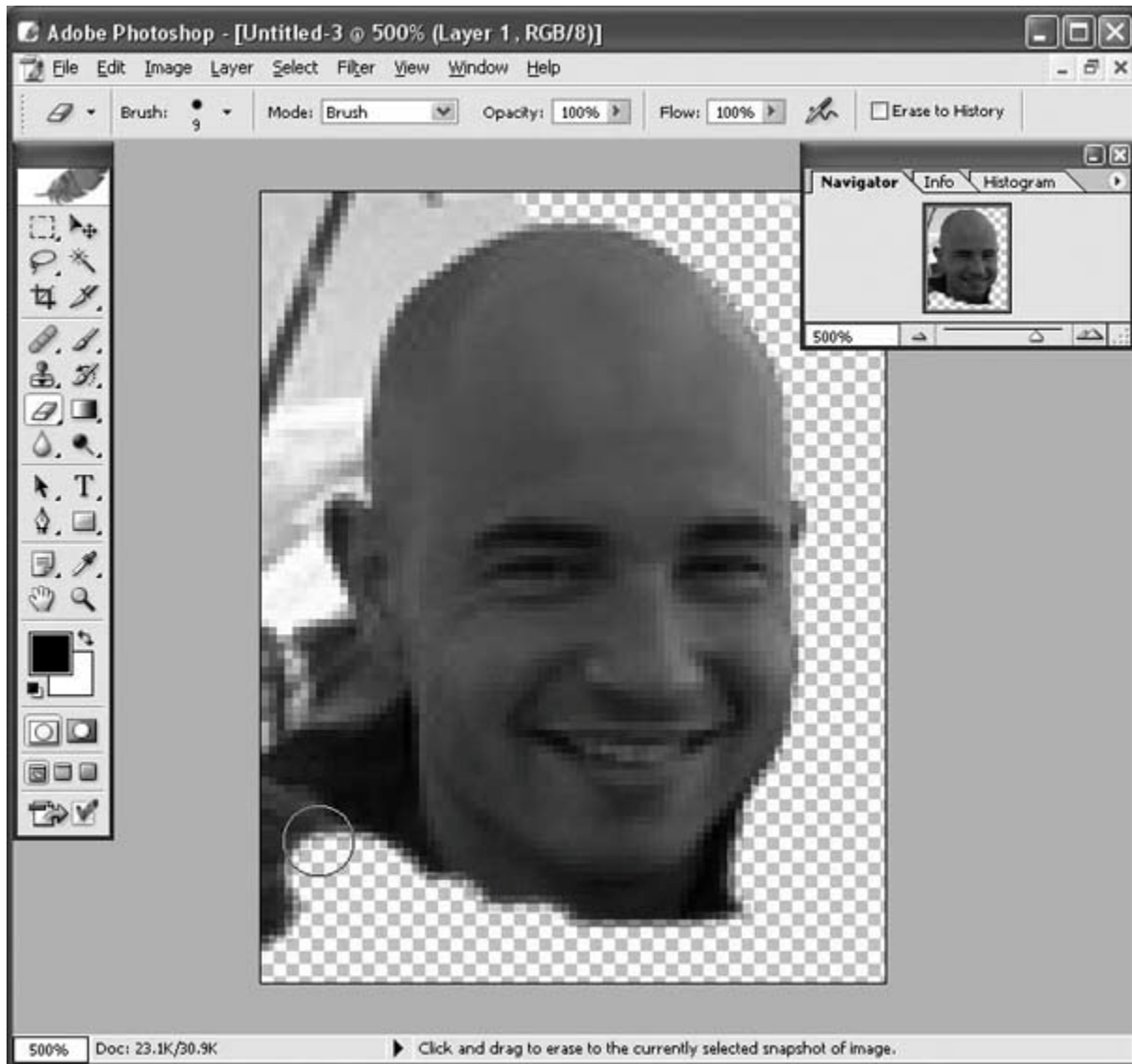
No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

2.6.2. Casting the Shadow

Now that you have the foreground image saved, you might want to show it on your map right away—but don't run ahead yet. We're going to want to reuse our work to create a shadow image. This is an image that gets placed behind the foreground image to give it that 3D effect, like it's sitting on top of the surface of your map. Using a shadow is optional, but it gives your custom markers more depth and character. This step is a little more complicated, but definitely worth your time.

Figure 2-11. Erasing around the head to create the foreground image



Here's the rundown, again from Adobe Photoshop:

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

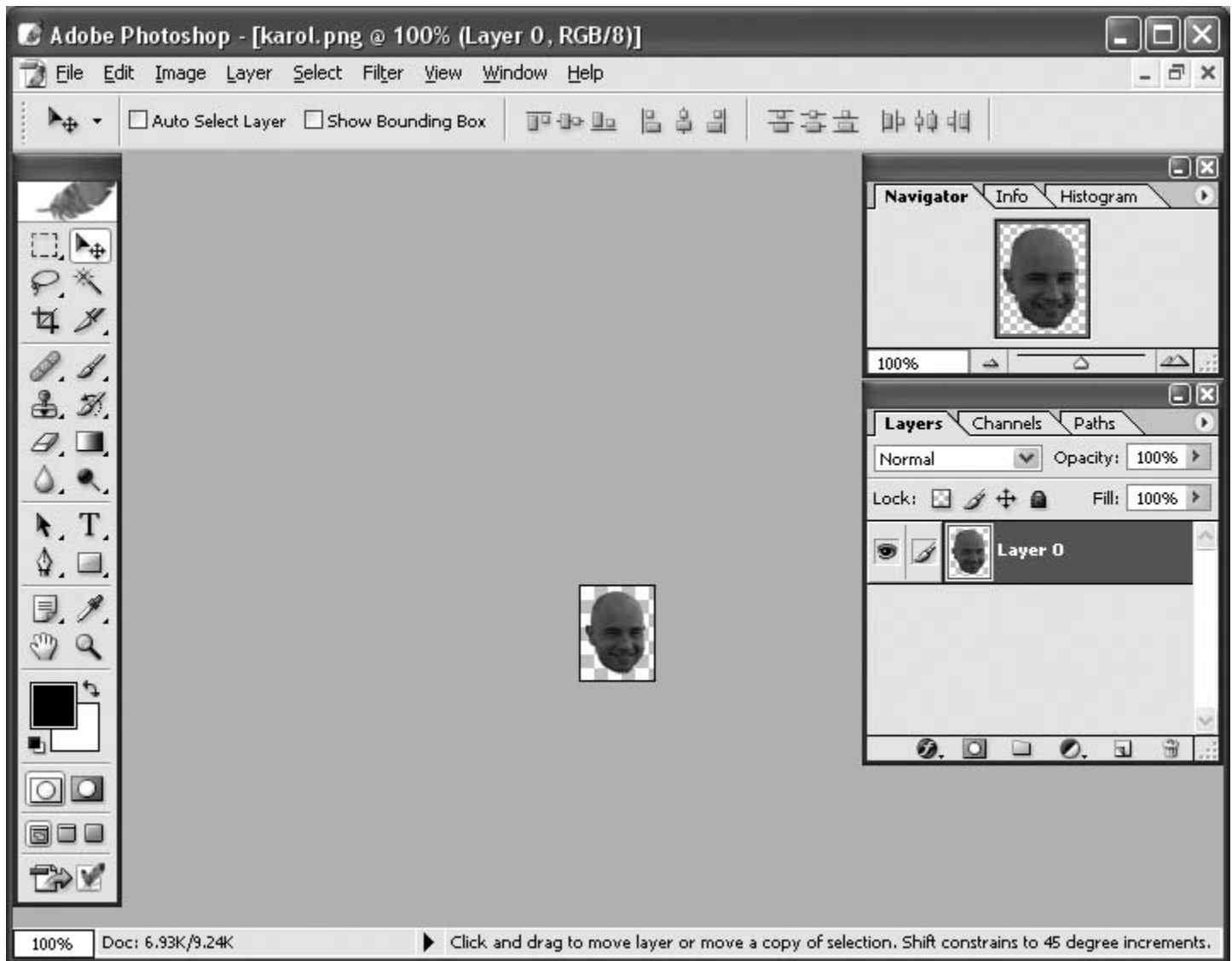
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

1. Image → Adjustments → Brightness/Contrast
Set both the Brightness and Contrast to -100.
2. Edit → Free Transform
Adjust the layer to exactly half its original height by grabbing the top line and dragging down.
3. Edit → Transform → Skew
Grab the top line and drag to the right to skew the image 45 degrees.

Figure 2-12. The final version of the foreground image



4. Image → Canvas Size
Adjust the canvas so that it fits your new drop shadow. Give it a bit of extra room so nothing gets cut off.
5. Filters → Blur → Gaussian Blur
This will give your shadow that fuzzy look. Try setting it to 0.9.
6. Layer → Layer Style → Blending Options → Fill Opacity

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Make the shadow transparent, so it doesn't look like a big black blob. Set the Fill Opacity to about 50%.

7. File → Save for Web

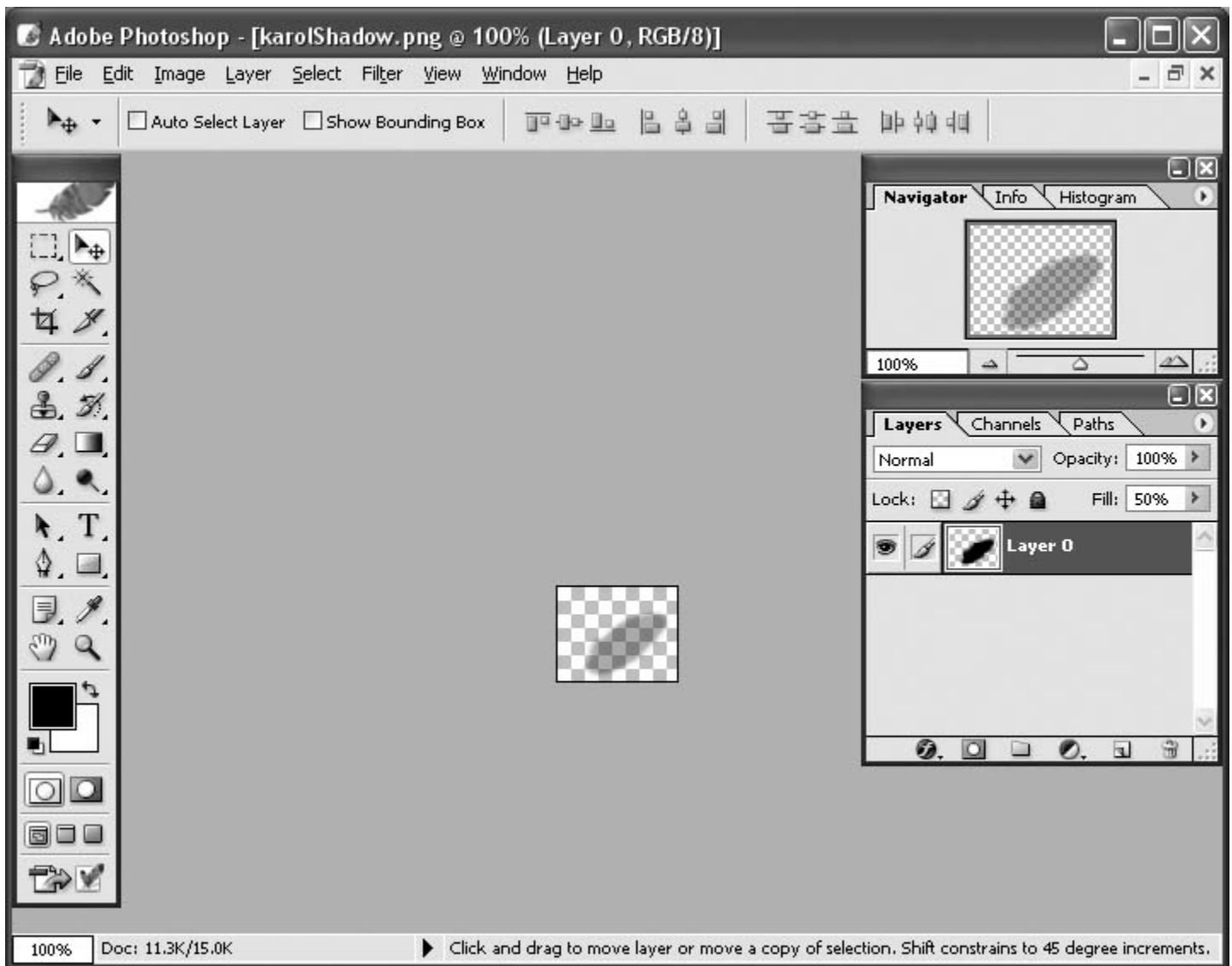
Remember to keep it a 24-bit PNG file with Transparency.

Figure 2-13 shows the finished shadow layer for the icon.

2.6.3. Add Your New Icon to a Map

Now that the two source images have been created, let's add them to your map. If we were making a generic marker, we would create an instance of the `GMarker` class, using a `GPoint` object as an argument. To create a custom marker on the other hand, we need to add an additional argument to the `GMarker` constructor, a `GIcon` object. Here is an example of how to use the `GIcon` constructor:

Figure 2-13. The head has a shadowy background



```

var icon = new GIcon();
icon.image = "http://igargoye.com/mapfiles/karol.png";
icon.shadow = "http://igargoye.com/mapfiles/karolShadow.png";
icon.iconSize = new GSize(43, 55);
icon.shadowSize = new GSize(70, 55);
icon.iconAnchor = new GPoint(0, 0);
icon.infoWindowAnchor = new GPoint(9, 2);
icon.infoShadowAnchor = new GPoint(18, 25);

```

The `GSize` object holds the size of your image. In this case, `icon.image` has a width of 43 pixels and a height of 55 pixels. The corresponding `icon.shadow` has a width of 70 pixels and is 55 pixels high. Specifying these image dimensions are critical, because if you don't, your photo will end up being distorted.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

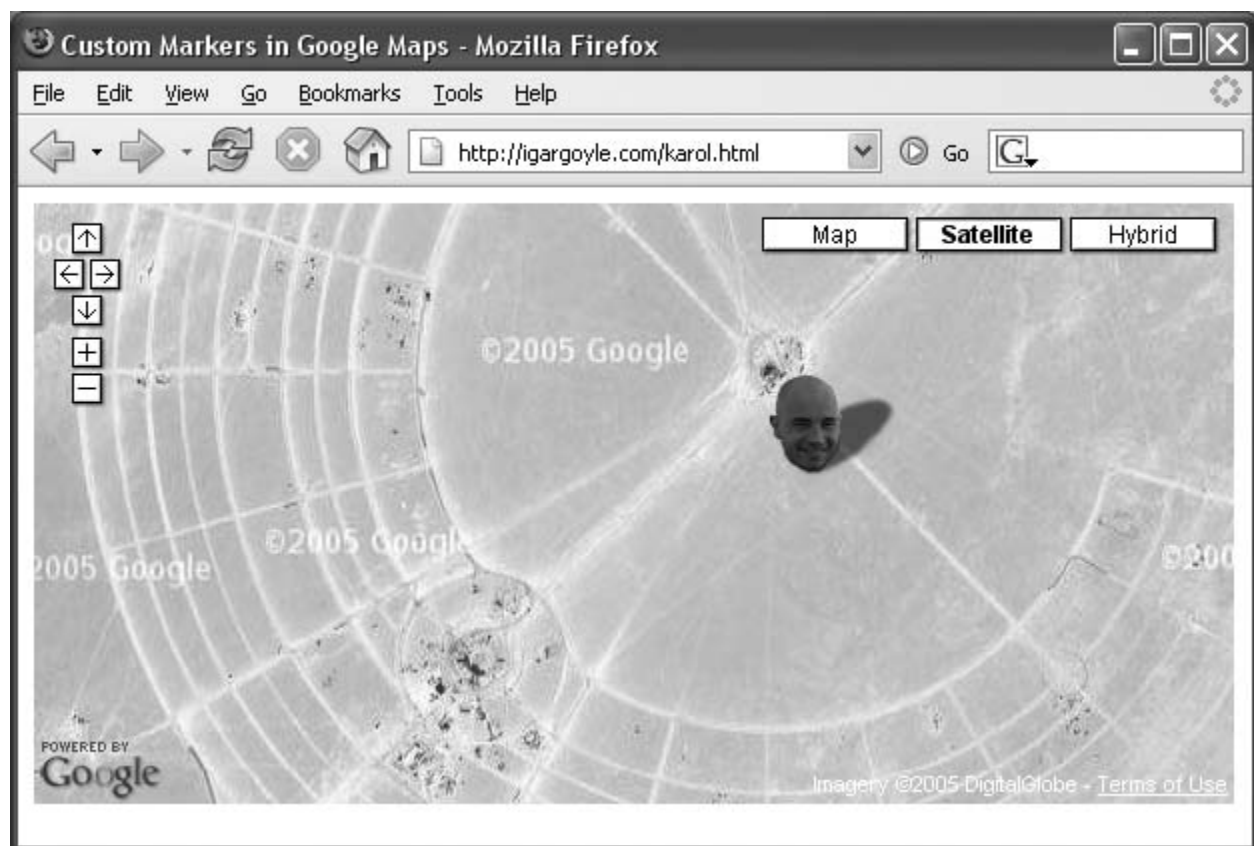
The `iconAnchor` property stores the point on the icon, relative to the top left corner of our icon image, where it should be anchored to the map. The `infoWindowAnchor` property is similar, except it specifies the anchor point on the image of the information window. Finally, the `infoShadowAnchor` indicates where the drop shadow below the info window should be anchored under your marker. If your icon is shaped like the standard Ride Finder icons, you can leave it as it is; otherwise, you should use your image editor to figure out where these points lie on your custom icon.

Finally, to add this to a new marker, you need to use the `GMarker` constructor with the `GIcon` as an extra argument.

```
var marker = new GMarker(point, icon);
map.addOverlay(marker);
```

Figure 2-14 shows our custom map icon on a satellite map showing Burning Man.

Figure 2-14. The disembodied flying heads all swarm to Burning Man



So now that you know how to create an icon with both a foreground image and a shadow, break out your artistic skills and make your map uniquely different. You can make it look professional, stylish, or even silly, like I did. The message *your* map communicates will be all the stronger for it.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

2.6.4. See Also

- Find some good, free, generic icons at <http://brennan.offwhite.net/blog/archives/000211.html>.

—Tom Longson

Hack 15. Map a Slideshow of Your Travels



Show your friends and family not just what you saw on your vacation, but where you saw it.

Wouldn't it be cool to be able to show your friends and family not only what you saw on your vacation, but also where each photo was taken? Thanks to Google Maps' simple API, you can very easily make a slideshow to put on your web site.

Travelogue slideshows are as old as the film slide projector itself. The Google Maps API and a dab of JavaScript can dust off the ancient tradition, and show off the places you've been with an amazing amount of detail. What's more, little touches like the API's animated map panning function lend a sense of motion to the story of your travels. [Figure 2-15](#) shows a slideshow I put together of my vacation last year to the Burning Man Festival in Black Rock City, Nevada. This slideshow can also be seen online at <http://igargoyle.com/slideshow.html>.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-15. Make a slideshow that will captivate your friends and family



For my map, I took the photos from Black Rock City and added titles and descriptions for each image to better tell the story of my trip. Since my relatives live on the other side of the world, this makes it very easy for me to update them about what's going on in my life—and show off photos without having to lob off enormous emails, stuffed with unreadable attachments.

To get started, you're going to need photos, each with its own latitude and longitude coordinates. The easiest way to do this is by bringing along a GPS when you photograph, and mark each image with a waypoint or write the location on a notepad. If you're interested in doing large batches of photos, pick up a copy of O'Reilly's *Mapping Hacks* and learn more about cross-referencing GPX tracklogs with the EXIF timestamp of digital photographs. Alternatively, you can adapt the technique described in "Where Did the User Click?" [[Hack #11](#)], and use your memory and a few map clicks to find the coordinates of where your photos were taken.

2.7.1. The Code

However you assemble the metadata for your photographs, you'll wind up with a JavaScript object that probably looks like this:

```
var myObj = {"photos": [
  {"img": "http://igargoyle.com/theman.jpg",
    "longitude": "-119.236402821", "latitude": "40.754786911",
    "title": "Black Rock City: 2005",
```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

```

        "description": "A Playa Slideshow using Google Maps. Just email
this to your friends, and when it loads, it will start playing
automatically."},
        {"img": "http://igargoyle.com/hammockhangout.jpg",
         "longitude": "-119.233165", "latitude": "40.7590351",
         "title": "The Hammock Hangout",
         "description": "A GIANT tent that uses passive solar principles to
stay cool during the day. Housed 50 hammocks, and had room for people to set
up their own too."},
        {"img": "http://igargoyle.com/us.jpg",
         "longitude": "-119.246943567", "latitude": "40.752424806",
         "title": "Some of the gang in camp ROAMnet",
         "description": "Karl, Jay, Jana, and Nym."},
    ]
};

```

To set up the timers for these slides, we can access the data array like so:

```

for (i = 0; i < myObj.photos.length; i++) {
    img = myObj.photos[i].img;
    longitude = myObj.photos[i].longitude;
    latitude = myObj.photos[i].latitude;
    title = myObj.photos[i].title;
    description = myObj.photos[i].description;

    loadPhoto(img, longitude, latitude, i, title, description);
}

```

The `loadPhoto` function takes these arguments and creates an anonymous function that calls the browser's built-in `window.setTimeout` function. This is the most important part of the hack because it tells the function to run at a certain time, specified in milliseconds. In this function, `10000*time` calculates the time for each photo, so the first photo loads immediately, the second photo loads after 10 seconds, the third photo loads after 20 seconds, and so on.

```

function loadPhoto(photoURL, longitude, latitude, time, title, description)
{
    // A simple timer, which delays the creation of the new
    // marker, changes the photo, and recenters the map.
    window.setTimeout( function() {
        // Create and place a marker for the photograph's location
        var marker = new GMarker(new GPoint(longitude, latitude));
        map.addOverlay(marker);

        // Change the titleBox and descriptionBox to reflect the
        // new photo's title and description
        document.getElementById("titleBox").innerHTML = title;
        document.getElementById("descriptionBox").innerHTML = description;

        // Change the src location of the photo element to the new location.
        document.getElementById("photo").src = photoURL;

        // Have our Google Map recenter or pan to the new location
        map.recenterOrPanToLatLng(new GPoint(longitude, latitude));

    }, 10000 * time); // Change 10000 to speed up or slow down the
    slideshow.
}

```

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

The function inside the `setTimeout` does all the real work, though. To begin with, it creates the marker for your photo using the `GMarker` constructor, and then a call to the map's `addOverlay` function. Secondly, this function displays the title and description on the page by calling each container and setting its respective `innerHTML` properties. Next, the function changes the source location of the only `img` element on our page to that of the current photo. Once all this is done, the map is re-centered on the new marker by using a call to the map's `recenterOrPanToLatLng` method. In my example, I made sure the points were close enough together to cause the map to pan instead of re-center, because it's great to see the map glide from one location to another.

A slideshow is a wonderful way to captivate your audience and tell a story. So get out there and make something truly wonderful that your family and friends will remember for years to come!

2.7.2. See Also

- The slideshow concept can also be integrated with photos stored on Flickr [[Hacks #47](#) and [#48](#)].

—Tom Longson

Hack 16. How Big Is the World?



If you wanted to make your own Google Maps server, how much hard drive space would you need?

Google Maps renders maps by stitching small images together. We seek to discover the storage capacity of such an image repository. By capturing and examining screenshots of Google Maps in action, we can estimate the map scale at each zoom level, which will give us an idea of how much space is necessary to store all the tiles for that zoom level. Finally, we can add the storage requirements for each zoom level and apply some simple rules of thumb to arrive at an idea of how much hard drive space is necessary to support a web mapping service such as Google Maps.

2.8.1. Economies of Scale

First, we need to discover the scaling factors used at each of the fifteen zoom steps. To accomplish this analysis, we use a tool called Art Director's Toolkit, which comes bundled with Mac OS X and which offers an overlay desktop ruler image for measuring pixel distances onscreen. In zoom levels 0 to 6, we measure the pixel length between the northeast corner of Colorado and the southeast corner of Wyoming. This distance is clearly marked on the map as a horizontal line, which makes measuring it easy. [Figure 2-16](#) depicts zoom levels 0, 1, and 2, where the distances in question are 12, 24, and 48 pixels, respectively.

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-16. Zoom levels 0 through 2



In Figure 2-17, we see that, for zoom levels 3, 4, and 5, the same distances are 98, 196, and 394 pixels.

For zoom level 6, the distance between the northeast corner of Colorado and the southeast corner of Wyoming measures out at 790 pixels. Zoom level 7 was skipped because there was nothing to measure for it—smaller things were too small, and bigger things were too big. (Skipping it did not negatively impact the analysis.)

Figure 2-17. Zoom levels 3 through 5



In zoom levels 8 through 14, we measure the pixel length of the path from the intersection of Trenton Street and East 16th Avenue to the intersection of Verbena Street and East 16th Avenue in Denver, Colorado, which is within the metropolitan area closest to our previous locations. For zoom level 8, the distance is 9 pixels. For zoom levels 9, 10, and 11, the distances are 19, 37, and 74 pixels. The results are shown in Figure 2-18.

Chapter 2. Introducing the Google Maps API

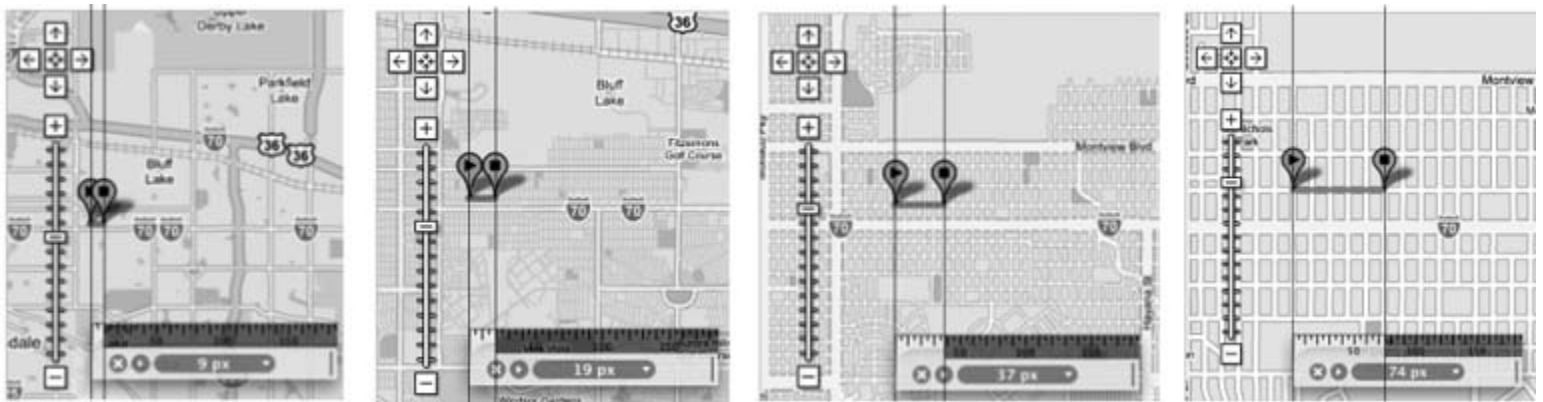
Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

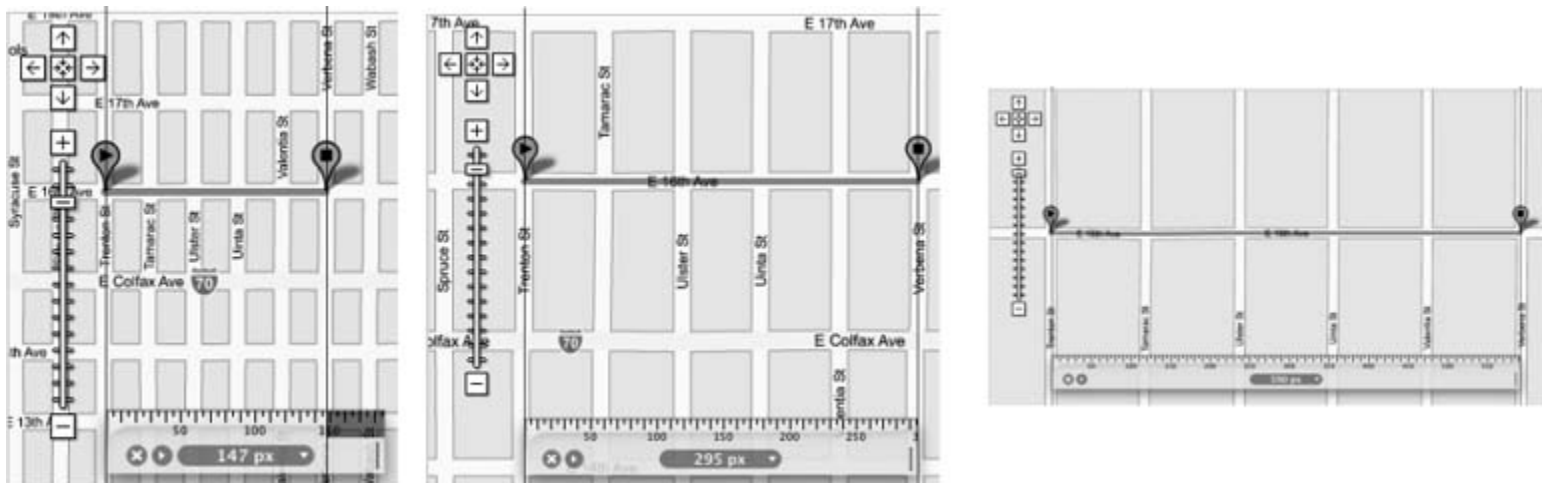
Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-18. Zoom levels 8 through 11



For zoom levels 12, 13, and 14, the distances are 147, 295, and 590 pixels. Figure 2-19 depicts this measurement.

Figure 2-19. Zoom levels 12 through 14



Now we can take the information from these measurements, and attempt to establish the numeric scale ratio between one zoom level and the previous one. Figure 2-20 presents the same relationships in three nicely formatted line graphs and Table 2-1 summarizes the data we collected.

Chapter 2. Introducing the Google Maps API

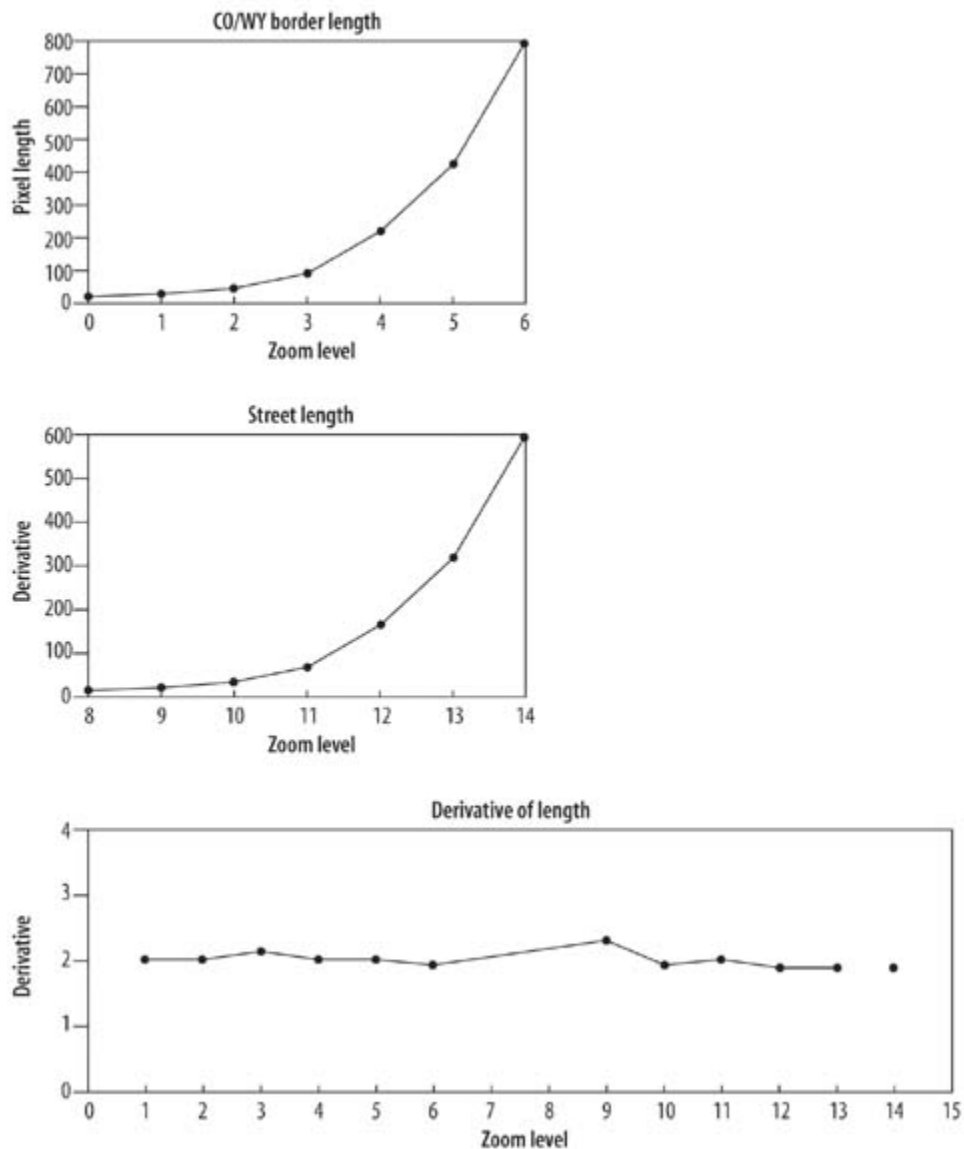
Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Figure 2-20. Length ratios visualized in a series of line graphs



The conclusion we draw is that we can be fairly certain that the scale doubles with every increment of the zoom bar.

Table 2-1. Length ratios from one zoom level to the previous zoom level

Zoom	State border length	Ratio	Zoom	Street length	Ratio
0	12	n/a	8	8	n/a
1	24	2	9	19	2.38

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Zoom	State border length	Ratio	Zoom	Street length	Ratio
2	48	2	10	37	1.95
3	98	2.04	11	74	2
4	196	2	12	147	1.99
5	394	2	13	295	2.01
6	790	2.01	14	590	2

2.8.2. So, How Much?

By zooming almost all the way out in Google Maps, we see that North America fits nicely in a 600 x 800–pixel rectangular region, amounting to 480,000 pixels. Armed with this approximation, we proceed to estimate the pixel-area of this body at each zoom level. [Table 2-2](#) depicts these relationships.

Table 2-2. Approximate area in pixels of North America for each zoom level

Zoom	Scale	Width	Height	Area in pixels
0	1	800	600	480,000
1	2	1,600	1,200	1,920,000
2	4	3,200	2,400	7,680,000
3	8	6,400	4,800	30,720,000
4	16	12,800	9,600	122,880,000
5	32	25,600	19,200	491,520,000
6	64	51,200	38,400	1,966,080,000
7	128	102,400	76,800	7,864,320,000
8	256	204,800	153,600	31,457,280,000

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.

Zoom	Scale	Width	Height	Area in pixels
9	512	409,600	307,200	125,829,120,000
10	1,024	819,200	614,400	503,316,480,000
11	2,048	1,638,400	1,228,800	2,013,265,920,000
12	4,096	3,276,800	2,457,600	8,053,063,680,000
13	8,192	6,553,600	4,915,200	32,212,254,720,000
14	16,384	13,107,200	9,830,400	128,849,018,880,000

If we add up the areas, we find that 171,798,691,680,000 (171 trillion) pixels are needed to store all the bitmap information. Since all maps are made up of 256 x 256 tiles, one can venture to guess that there are $171,798,691,680,000 \div (256 \times 256) = 2,621,439,997$ (2.6 billion) potential tile files.

The color histogram of the maps in [Figure 2-19](#) shows that about 60 percent of it is water. Assuming that Google observes such statistics, we guess that a single tile is used for all water regions. There are also lots of regions (such as tundra, deserts, and forests) where uniformly colored tiles can be used. Computing this accurately is difficult, but we will say it amounts to 10 percent of the data. So, only 30 to 40 percent of the tiles have unique data on them. This reduces the amount of data to 50 to 70 trillion raw data pixels stored in 750 million to 1 billion image files. Assuming a modest 1 byte per 6 pixels compression ratio (for LZW-encoded GIF format images), the storage required might be 50 to 70 trillion pixels * (1 byte/6 pixels) = 8 to 11 terabytes. If we consider that Google supports three map types at present (Map, Satellite, and Hybrid), this suggests that 24 to 33 terabytes are needed to store all the image data.

2.8.3. What About the Rest of the World?

Since we did our original analysis, Google Maps UK, Google Maps Japan, and Google Earth were introduced, providing further evidence of a lofty goal to create a world atlas. So this puny analysis (as compared to the world's topology and architectural landmark data necessary for Google Earth), makes an attempt at covering the whole earth with tiles. To do this, we must learn more about the world. The *CIA World Factbook* provides just what we need.

To wrap the world requires 510 million km² of surface. Of this, only 29.2%, or 147 million square kilometers, is land. North America's surface area is about 21.4 million square kilometers (9.9 for Canada, 9.6 for the United States, and 1.9 for Mexico) or 13.6% of the world's total land surface area.

We concluded from our analysis that covering North America requires somewhere between 750,000 and 1 billion distinct tiles to be fully described. Now we know that this is only 13.6% of the tiles necessary to describe the world's land tiles. So, anywhere from 5.5 to 7 billion distinct tiles ought to cover the world's surface area. Assuming the compression ratio described above, the world's tiles amount to 61 to 81 terabytes just for the rendered vector maps, and *182 to 243 terabytes* for all three map types. That's a lot of data—but then storing and retrieving huge amounts of data is Google's stock in trade!

Chapter 2. Introducing the Google Maps API

Google Maps Hacks By Schuyler Erle, Rich Gibson

ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Prepared for Douglas Looms, Safari ID: dlooms@erols.com, User number: 328147
Copyright 2006, Safari Books Online, LLC.



Since this was written, Google has added three more zoom levels to Google Maps, for a total of 18! The extra math is left as an exercise for the interested reader.

In some ways, it seems a bit comical to attempt such a calculation where every step of the way requires an approximation. That's why in the end we have such a wide chasm of error. And, of course, this rough analysis does not cover area distortion introduced by mapping the globe's points onto a two dimensional surface. However, even with this rough estimate, we think we've managed to get a decent sense of just what it takes to map the entire world in the style that Google Maps has pioneered.

—*Michal Guerquin and Zach Frazier*