Beginning Google Maps Applications with Rails and Ajax

From Novice to Professional

Andre Lewis, Michael Purvis, Jeffrey Sambells, and Cameron Turner

Apress®

Beginning Google Maps Applications with Rails and Ajax: From Novice to Professional

ISBN-13 (pbk): 978-1-59059-787-3

ISBN-10 (pbk): 1-59059-787-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jason Gilmore
Technical Reviewer: Sam Aaron
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Matt Wade
Project Manager: Sofia Marchant
Copy Edit Manager: Nicole Flores
Copy Editor: Jennifer Whipple
Assistant Production Director: Kari Brooks-Copony
Production Editor: Laura Cheu
Compositor: Kinetic Publishing Services, LLC
Proofreader: April Eddy
Indexer: Beth Palmer
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at the official web site, `http://googlemapsbook.com`.

Contents at a Glance

About the Authors xiv
About the Technical Reviewer xvi

PART 1 Your First Google Maps

CHAPTER 1 Google Maps and Rails 3
CHAPTER 2 Getting Started 13
CHAPTER 3 Interacting with the User and the Server 33
CHAPTER 4 Geocoding Addresses 69

PART 2 Beyond the Basics

CHAPTER 5 Manipulating Third-Party Data 99
CHAPTER 6 Improving the User Interface 123
CHAPTER 7 Optimizing and Scaling for Large Data Sets 147
CHAPTER 8 What's Next for the Google Maps API? 197

PART 3 Advanced Map Features and Methods

CHAPTER 9 Advanced Tips and Tricks 207
CHAPTER 10 Lines, Lengths, and Areas 261
CHAPTER 11 Advanced Geocoding Topics 287

PART 4 Appendixes

APPENDIX A Finding the Data You Want 315
APPENDIX B Google Maps API 323

INDEX 357

Contents

About the Authors xiv
About the Technical Reviewer xvi

PART 1 Your First Google Maps

CHAPTER 1 Google Maps and Rails 3

KML: Your First Map 4
Wayfaring: Your Second Map 5
Adding the First Point 6
Adding the Flight Route 7
Adding the Destination Point 8
Adding a Driving Route 9
Got Rails? 10
What's Next? 11

CHAPTER 2 Getting Started 13

On JavaScript, Helpers, and Plug-ins 13
Creating Your Rails Application 14
The First Map 14
Keying Up 14
Examining the Sample Map 17
Specifying a New Location 18
Separating Code from Content 20
Cleaning Up 22
Basic User Interaction 23
Using Map Control Widgets 23
Creating Markers 24
Detecting Marker Clicks 26
Opening the Info Window 27

A List of Points 28
Using Arrays and Objects 28
Iterating 30
Summary 32

CHAPTER 3 Interacting with the User and the Server 33

Adding Interactivity 33
Going on a Treasure Hunt 34
Reviewing Application Structure 35
Building on Your Application 36
Creating a New Controller 36
Creating a Marker Model and Migration 36
Creating the Database, Connecting via Rails, and Running the Migration 37
Creating the Map View 38
Creating the Map and Marking Points 38
Listening to User Events 39
Asking for More Information with an Info Window 42
Creating an Info Window on the Map 43
Embedding a Form into the Info Window 44
Avoiding an Ambiguous State 48
Controlling the Info Window Size 50
Implementing Ajax 52
Google's GXmlHttp vs. Prototype's Ajax.Request 52
Using Google's Ajax Object 53
Saving Data with GXmlHttp 53
Parsing the JSON Structure 58
Retrieving Markers from the Server 59
Adding Some Flair 62
Ajax with Prototype 65
Summary 67

CHAPTER 4 Geocoding Addresses 69

Preparing the Address Data 69
Creating the Model 70
Adding a full_address Method 71
Populating the Table 71
Using Geocoding Web Services 73
Requirements for Consuming Geocoding Services 73
The Google Maps API Geocoder 74

The Google JavaScript Geocoder 81
The Yahoo Geocoding API 82
Geocoder.us 87
Geocoder.ca 89
Services for Geocoding Addresses Outside Google's Coverage 91
Persisting Lookups 92
Building a Store Location Map 93
Summary 96

PART 2 Beyond the Basics

CHAPTER 5 Manipulating Third-Party Data 99

Using Downloadable Text Files 99
Downloading the Database 100
Working with Files 103
Correlating and Importing the Data 104
Using Your New Database Schema 107
Screen Scraping 115
Our Scraping Tool: scrAPI 116
Screen Scraping Considerations 121
Summary 121

CHAPTER 6 Improving the User Interface 123

CSS: A Touch of Style 124
Maximizing Your Map 126
Adding Hovering Toolbars 128
Creating Collapsible Side Panels 130
Scripted Style 133
Switching Up the Body Classes 133
Resizing with the Power of JavaScript 135
Populating the Side Panel 138
Getting Side Panel Feedback 140
Data Point Filtering 141
RJS and Draggable Toolbars 144
RJS Templates and Partials 144
Draggable Toolbars 145
Summary 145

CHAPTER 7 Optimizing and Scaling for Large Data Sets 147

Understanding the Limitations 147
Streamlining Server-Client Communications 148
Optimizing Server-Side Processing 150
Server-Side Boundary Method 150
Server-Side Common-Point Method 155
Server-Side Clustering 160
Custom Detail Overlay Method 165
Custom Tile Method 174
Optimizing the Client-Side User Experience 182
Client-Side Boundary Method 183
Client-Side Closest-to-a-Common-Point Method 185
Client-Side Clustering 188
Further Client-Side Optimizations 192
Summary 194

CHAPTER 8 What's Next for the Google Maps API? 197

Driving Directions 197
Integrated Google Services 198
KML Data 200
More Data Layers 200
Beyond the Enterprise 202
Interface Improvements 202
Summary 204

PART 3 Advanced Map Features and Methods

CHAPTER 9 Advanced Tips and Tricks 207

Debugging Maps 207
Interacting with the Map from the API 208
Helping You Find Your Place 209
Force Triggering Events with GEvent 210
Creating Your Own Events 212
Creating Map Objects with GOverlay 212
Choosing the Pane for the Overlay 212
Creating a Quick Tool Tip Overlay 214

Creating Custom Controls 218
Creating the Control Object 219
Creating the Container 220
Positioning the Container 220
Using the Control 221
Adding Tabs to Info Windows 221
Creating a Tabbed Info Window 222
Gathering Info Window Information and Changing Tabs 224
Creating a Custom Info Window 224
Creating the Overlay Object and Containers 230
Drawing a LittleInfoWindow 231
Implementing Your Own Map Type, Tiles, and Projection 235
GMapType: Gluing It Together 236
GProjection: Locating Where Things Are 237
GTileLayer: Viewing Images 244
The Blue Marble Map: Putting It All Together 247
Summary 258

CHAPTER 10 Lines, Lengths, and Areas 261

Starting Flat 261
Lengths and Angles 262
Areas 263
Moving to Spheres 266
The Great Circle 267
Great-Circle Lengths 268
Area on a Spherical Surface 270
Working with Polylines 274
Building the Polylines Demo 275
Expanding the Polylines Demo 281
What About UTM Coordinates? 282
Running Afoul of the Date Line 284
Summary 285

CHAPTER 11 Advanced Geocoding Topics 287

Where Does the Data Come From? 287
Sample Data from Government Sources 288
Sources of Raw GIS Data 291
Geocoding Based on Postal Codes 292

Using the TIGER/Line Data 296
Understanding and Defining the Data 296
Parsing and Importing the Data 300
Building a Geocoding Service 307
Summary 312

PART 4 Appendixes

APPENDIX A Finding the Data You Want 315

Knowing What to Look For: Search Tips 315
Finding the Information 315
Specifying Search Terms 316
Watching for Errors 316
The Cat Came Back: Revisiting the TIGER/Line 316
Airports in TIGER/Line 318
The Government Standard: The GeoNames Data 319
Shake, Rattle, and Roll: The NOAA Goldmine 319
For the Space Aficionado in You 321
Crater Impacts 322
UFO/UAP Sightings 322

APPENDIX B Google Maps API 323

class GMap2 323
GMap2 Constructor 323
GMap2 Methods 324
class GMapOptions 329
GMapOptions Properties 329
enum GMapPane 330
GMapPane Constants 330
class GKeyboardHandler 330
GKeyboardHandler Bindings 330
GKeyboardHandler Constructor 331
interface GOverlay 331
GOverlay Constructor 331
GOverlay Static Method 331
GOverlay Abstract Methods 331
class GInfoWindow 332
GInfoWindow Methods 332
GInfoWindow Event 332

class GInfoWindowTab 333
GInfoWindowTab Constructor 333
class GInfoWindowOptions 333
GInfoWindowOptions Properties 333
class GMarker 333
GMarker Constructor 333
GMarker Methods 334
GMarker Events 335
class GMarkerOptions 335
GMarkerOptions Properties 335
class GPolyline 336
GPolyline Constructor 336
GPolyline Factory Methods 336
GPolyline Methods 336
GPolyline Event 336
class GIcon 337
GIcon Constructor 337
GIcon Constant 337
GIcon Properties 337
class GPoint 338
GPoint Constructor 338
GPoint Properties 338
GPoint Methods 338
class GSize 338
GSize Constructor 338
GSize Properties 339
GSize Methods 339
class GBounds 339
GBounds Constructor 339
GBounds Properties 339
GBounds Methods 339
class GLatLng 340
GLatLng Constructor 340
GLatLng Properties 340
GLatLng Methods 340
class GLatLngBounds 341
GLatLngBounds Constructor 341
GLatLngBounds Methods 341
interface GControl 341
GControl Constructor 342
GControl Methods 342

class GControl 342
GControl Constructors 342
class GControlPosition 342
GControlPosition Constructor 343
enum GControlAnchor 343
GControlAnchor Constants 343
class GMapType 343
GMapType Constructor 343
GMapType Methods 343
GMapType Constants 344
GMapType Event 344
class GMapTypeOptions 344
GMapTypeOptions Properties 345
interface GTileLayer 345
GTileLayer Constructor 345
GTileLayer Methods 345
GTileLayer Event 346
class GCopyrightCollection 346
GCopyrightCollection Constructor 346
GCopyrightCollection Methods 346
GCopyrightCollection Event 346
class GCopyright 346
GCopyright Constructor 346
GCopyright Properties 347
interface GProjection 347
GProjection Methods 347
class GMercatorProjection 348
GMercatorProjection Constructor 348
GMercatorProjection Methods 348
namespace GEvent 348
GEvent Static Methods 348
GEvent Event 349
class GEventListener 349
namespace GXmlHttp 350
GXmlHttp Static Method 350
namespace GXml 350
GXml Static Methods 350
class GXslt 350
GXslt Static Methods 350

namespace GLog 350
GLog Static Methods 351
class GDraggableObject 351
GDraggableObject Static Methods 351
GDraggableObject Constructor 351
GDraggableObject Properties 351
GDraggableObject Methods 351
enum GGeoStatusCode 352
GGeoStatusCode Constants 352
enum GGeoAddressAccuracy 352
class GClientGeocoder 352
GClientGeocoder Constructor 352
GClientGeocoder Methods 353
class GGeocodeCache 353
GGeocodeCache Constructor 353
GGeocodeCache Methods 353
class GFactualGeocodeCache 354
GFactualGeocodeCache Constructor 354
GFactualGeocodeCache Method 354
class GMarkerManager 354
GMarkerManager Constructor 354
GMarkerManager Methods 354
GMarkerManager Events 355
class GMarkerManagerOptions 355
GMarkerManagerOptions Properties 355
Functions 355

INDEX 357

About the Authors

ANDRE LEWIS became interested in Google Maps when he set out to create a simple online list of local Wi-Fi cafes. That effort subsequently grew into an active community-driven site at `http://hotspotr.com`. Since then, he has developed numerous tools and techniques for map-based applications using Ruby on Rails.

While geographically oriented applications remain a favorite subject, Andre enjoys working with all kinds of technologies. He has architected systems to support millions of daily page views, but he also likes getting the JavaScript and CSS "just right" on a web page. He currently works freelance, consulting on Web 2.0 technologies and developing Ruby on Rails applications. He blogs at `http://earthcode.com` and speaks periodically at Bay Area technology groups.

Andre lives and works in San Francisco. When he's not working with clients or exploring the latest technologies, he likes to mountain bike, camp, and ride his motorcycle.

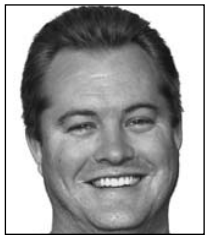
MICHAEL PURVIS is a mechatronics engineering student at the University of Waterloo, Ontario. Prior to discovering web scripting, he was busy with projects of other kinds, such as making a LEGO Mindstorms kit play the game Connect 4. After the PHP edition of this book was published, he was hired by Google for a four-month internship in the New York City office. He also continues to maintain an active community site for classmates, built from home-brewed extensions to PunBB and MediaWiki.

He has written about CSS for the Position Is Everything web site, and occasionally participates in the css-discuss mailing list. He loves simplicity, but cannot resist the charm of those few problems that do require negative margins and devious float tricks. Discussion of these and other nontechnical topics appears occasionally on his blog at `http://uwmike.com`.

Offline, he enjoys cooking, writing, cycling, and social dancing. He has worked with We-Create Inc. on a number of exciting PHP-based projects and has a strong interest in independent web standards.

JEFFREY SAMBELLS is a graphic designer and self-taught web applications developer best known for his unique ability to merge the visual world of graphics with the mental realm of code. With a bachelor of technology degree in graphic communications management and a minor in multimedia, Jeffrey was originally trained for the traditional paper-and-ink printing industry, but he soon realized the world of pixels and code was where his ideas would prosper. In 1999, he cofounded We-Create Inc., an Internet software company based in Waterloo, Ontario, which began many long nights of challenging and creative innovation. Currently, as director of research and development for We-Create, Jeffrey is responsible for investigating new and emerging Internet technologies and integrating them using web standards–compliant methods. In late 2005, he also became a Zend Certified Engineer.

When not playing at the office, Jeffrey enjoys a variety of hobbies, from photography to woodworking. When the opportunity arises, he also enjoys floating in a canoe on the lakes of Algonquin Provincial Park or going on an adventurous, map-free drive with his wife. Jeffrey also maintains a personal web site at `http://JeffreySambells.com`, where he shares thoughts, ideas, and opinions about web technologies, photography, design, and more. He lives in Ontario, Canada, eh, with his wife, Stephanie, daughter, Addison, and their little dog, Milo.



CAMERON TURNER has been programming computers since his first VIC 20 at age seven. He has been developing interactive web sites since 1994. In 1999, he cofounded We-Create Inc., which specializes in Internet software development. He is now the company's chief technology officer. Cam obtained his honors degree in computer science from the University of Waterloo with specialization in applied cryptography, database design, and computer security.

Since the PHP edition of this book was published, Cam started giving technology-related talks and lectures to companies and associations based in and around Waterloo, Ontario. Topics and interests range from Google Maps (of course) to search engine optimization as well as other topics relating to professional web software development.

Cam lives in Canada's technology capital of Waterloo with his wife, Tanya, son, Owen, and dog, Katie. His hobbies include geocaching, biking, hiking, water skiing, and painting. He maintains a low-volume personal blog at `http://CamTurner.com`, discussing nontechnical topics, thoughts, theories, and family life.

About the Technical Reviewer

SAM AARON is a Ph.D. student at the School of Computing Science at Newcastle University in the U.K. He is currently finishing off his thesis on the subject of interest management. He is both a Ruby and a Rails fanatic, and as such is actively involved in using and raising awareness of these wonderful technologies. He founded and organizes the local Ruby and Rails User Group—ncl.rb, which attracts more than 20 people every month. He is using Rails to build a web-based decision support tool for Newcastle University's transport department and is constantly looking for excuses to include as many of the exciting new Rails advances into his projects as possible.

Sam lives with his beautiful girlfriend, Susanna, on the quayside in Newcastle-upon-Tyne, England. He loves watching the birds fly over the river from his window. He spends his working days hacking away on his PowerBook listening to strange electronic music. When not working, he likes to get out of the city and relax. He loves camping, climbing mountains, and power kiting. When at home he's often found either playing the piano or table football. He doesn't own a car and can't even drive, preferring instead to cycle everywhere—especially long-distance expeditions with good friends.

When Sam finishes his Ph.D., he plans to start a web development, training, and consultancy company focusing on Ruby and Rails. He is currently in talks with Newcastle College with respect to it including Ruby and Rails content within its taught courses. If you're interested in finding out what Sam's up to today, just head along to his blog: `http://sam.aaron.name`.

PART 1

Your First Google Maps

CHAPTER 1

Google Maps and Rails

The last year or so has been an incredibly exciting time for web developers. New tools have come out that make web development easier, more productive, and more fun. A slew of new APIs are available that let you glue together data and services in interesting ways. As developers, we are more empowered with new and interesting technologies than ever before.

This book focuses on one API that has had a particularly profound impact: the Google Maps API. Since you have this book in hand, you're probably already convinced of Google Maps' importance. In case you need a reminder, however, visit Google Maps Mania (`http://googlemapsmania.blogspot.com`) for a view into the sprawling culture of innovation that Google Maps has fostered in the development community. The Google Maps API has spawned a whole class of web-based applications that would have been impossible to create without it.

You're going to use the Google Maps API on a platform that has inspired an equally fervent following: Ruby on Rails. The Rails framework facilitates radical improvements in productivity within its niche: database-backed web applications. Rails is intuitive, powerful, and free. Together, Rails and Google Maps enable you, the developer, to build impressive web-based applications that would have been difficult or impossible two years ago.

Over the course of the coming chapters, you're going to move from simple tasks involving markers and geocoding to more advanced topics, such as how to acquire data, present many data points, and provide a useful and attractive user interface.

There are many reasons why Ruby on Rails is an ideal platform to work with Google Maps. Rails makes it trivial to produce and consume XML, which Google Maps uses extensively. Rails also has built-in support for JSON (JavaScript Object Notation), a concise format for passing structured data from server to browser. Finally, Ruby has some excellent libraries for screen-scraping, which we will employ in later chapters.

We are assuming that you are coming to this book with a certain amount of Rails experience. You probably already have Ruby and Rails installed in a development environment and know how to get an application up and running. If not, don't fear: we list some resources in the sidebar "Just Getting Started with Ruby and Rails?" near the end of this chapter to help you get rolling on Rails. Whatever your current skill level vis-à-vis Rails, this book will get you in the mapping game and tell you everything you need to create killer maps applications. With the power of the Rails framework, the Ruby language, and the Google Maps API, you will command a development toolkit to be reckoned with.

We know you're eager to get started on a map project, but before we dig into the code, we want to show you two simple ways of creating ultraquickie maps: with KML (Keyhole Markup Language) files and through the Wayfaring map site.

Both these approaches are stepping stones; we will use them as easy introductions to the world of Google Maps. In Chapter 2, you will begin digging into code, which will of course lead to much greater flexibility and sophistication in what you can build.

KML: Your First Map

KML is one of the easiest methods to get your own markers and content displayed on a Google map. As you would expect, the ease is at the expense of flexibility, but it's still a great way to get started. In June 2006, Google announced that its official maps site would support the plotting of KML files. You can simply plug a URL into the search box, and Google Maps will show whatever locations are contained in the KML file specified by the URL. We aren't going to go in depth on this, but we've made a quick example to show you how powerful the KML method is, even if it is simple.

■ **Note** The name *Keyhole Markup Language* is a nod to both its XML structure and Google Earth's heritage as an application called Keyhole. Keyhole was acquired by Google in late 2004.

We created a file called `toronto.kml` and placed the contents of Listing 1-1 in it. The paragraph blurbs are borrowed from Wikipedia, and the coordinates were discovered by manually finding the locations on Google Maps.

Listing 1-1. *A Sample KML File*

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.google.com/earth/kml/2">
<Document>
  <name>toronto.kml</name>
  <Placemark>
    <name>CN Tower</name>
    <description> The CN Tower (Canada's National Tower, Canadian National Tower),
    at 553.33 metres (1,815 ft., 5 inches) is the tallest ➥
    freestanding structure on land.
    It is located in the city of Toronto, Ontario, Canada, and is considered the
    signature icon of the city. The CN Tower attracts close to two million visitors
    annually.

    http://en.wikipedia.org/wiki/CN_Tower</description>
    <Point>
      <coordinates>-79.386864,43.642426</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

In the actual file (located at `http://book.earthcode.com/kml/toronto.kml`), we included two more `Placemark` elements that point to other well-known buildings in Toronto. To view this on Google Maps, paste the previous URL into the Google Maps search field. Alternatively, you can just visit the following link: `http://maps.google.com/maps?f=q&hl=en&q=http://book.earthcode.com/kml/toronto.kml`.

Figure 1-1 shows what it looks like.

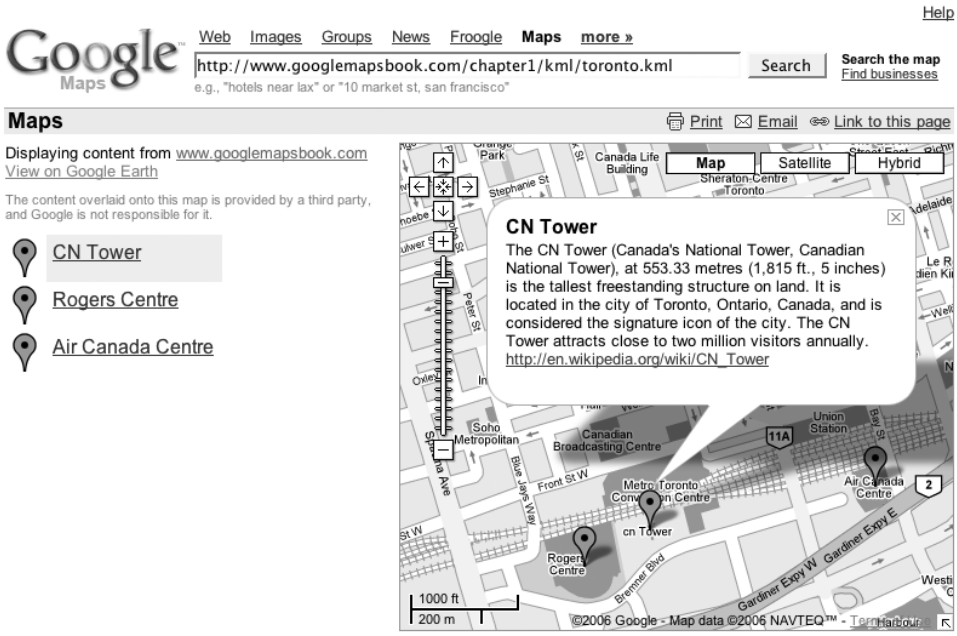


Figure 1-1. *A custom KML data file being displayed at maps.google.com*

Now, is that a quick result or what? Indeed, if all you need to do is show a bunch of locations, it's possible that a KML file will serve your purpose. If you're trying to link to your favorite fishing spots, you could make up a KML file, host it somewhere for free, and be finished.

But that wouldn't be any fun, would it? After all, as cool as the KML mapping is, it doesn't actually offer any interactivity to the user. In fact, most of the examples you'll work through in Chapter 2 are just replicating the functionality that Google provides here out of the box. But once you get to Chapter 3, you'll start to see things that you can do *only* when you harness the full power of the Google Maps API.

Before moving on, though, we'll take a look at one other way of getting a map online quickly.

Wayfaring: Your Second Map

A number of services out there let you publish free maps of quick plotted-by-hand data. One of these, which we'll demonstrate here, is Wayfaring, shown in Figure 1-2. Wayfaring has received attention and praise for its classy design and community features (such as commenting and shared locations). Wayfaring is also built using Ruby on Rails.

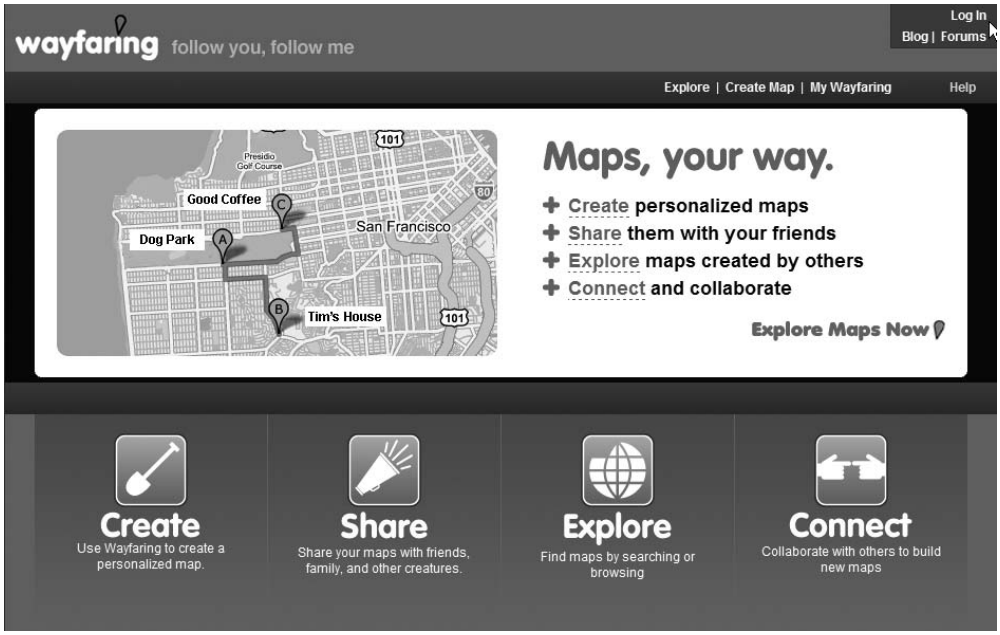


Figure 1-2. *Wayfaring home page*

Wayfaring is a mapping service that uses the Google Maps API and allows users to quickly create maps of anything they like. For example, some people make maps of their vacations; others have identified interesting aspects of their hometown or city. We'll walk you through making a quick map of an imaginary trip to the Googleplex in Mountain View, California.

Point your browser at `http://www.wayfaring.com` and follow the links to sign up for an account (clicking Log In will bring up the option to create a new account). Once you've created and activated your account, you can begin building your map by clicking the Create Map link in the upper right.

Adding the First Point

Let's start by adding the home airport for our imaginary journey. We're going to use Lester B. Pearson International Airport in Toronto, Ontario, Canada, but you could use the airport closest to you. Since Pearson is an international location (outside the United States), you need to drag and zoom the map view until you find it. If you're in the United States, you can use the nifty Jump To feature to search by text string. Figure 1-3 shows Pearson nicely centered and zoomed.

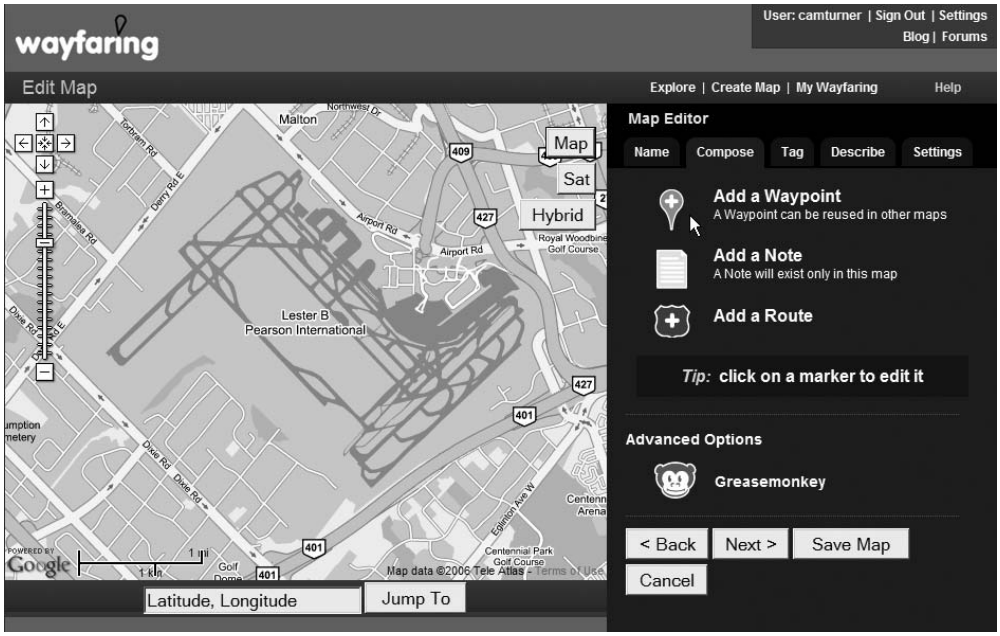


Figure 1-3. *Lester B. Pearson International Airport, Toronto, Ontario*

Once you've found your airport, you can click Next and name the map. Click Next again (after naming your map), and you should be back at the main Map Editor screen.

Select Add a waypoint from the list of options on the right. You'll be prompted to name the waypoint. We'll call ours *Lester B. Pearson International Airport*. However, as you type, you'll find that Wayfaring suggests this exact name. This means that someone else on some other map has already used this waypoint, and the system is giving you a choice of using their point or making one of your own. It's a safe bet that most of the airports you could fly from are already in Wayfaring, so feel free to use the suggested one if you would like. For the sake of the learning experience, let's quickly make our own. Click Next to continue.

The next two screens ask you to tag and describe this point in order to make your map more searchable for other members. We'll add the tags "airport Toronto Ontario Canada" and give it a simple description. Finally, click Done to commit the point to the map, which returns you to the Map Editor screen.

Adding the Flight Route

The next element you're going to add to your map is a *route*. A route is a line made up of as many points as you like. We'll use two routes in this example. The first will be a straight line between the two airports to get a rough idea of the distance the plane will have to travel to get us to Google's headquarters. The second will be used to plot the driving path we intend to take between the San Francisco airport and the Googleplex.

To begin, click Add a Route, name the route (something like *airplane trip*), and then click your airport. A small white dot appears on the place you click. This is the first point on your line. Now zoom out, scroll over to California, and zoom in on San Francisco. The airport is on the west side of the bay. Click the airport here, too. As you can see in Figure 1-4, a second white dot appears on the airport, and a blue line connects the two points. You can see the distance of the flight on the right side of the screen, underneath the route label. Wow, the flight seems to have been more than 2,000 miles! If you make a mistake and accidentally click on the map a few extra times (thereby creating extraneous midway points) in the process of getting to San Francisco, you can use the Undo Last option. Otherwise, click Save.

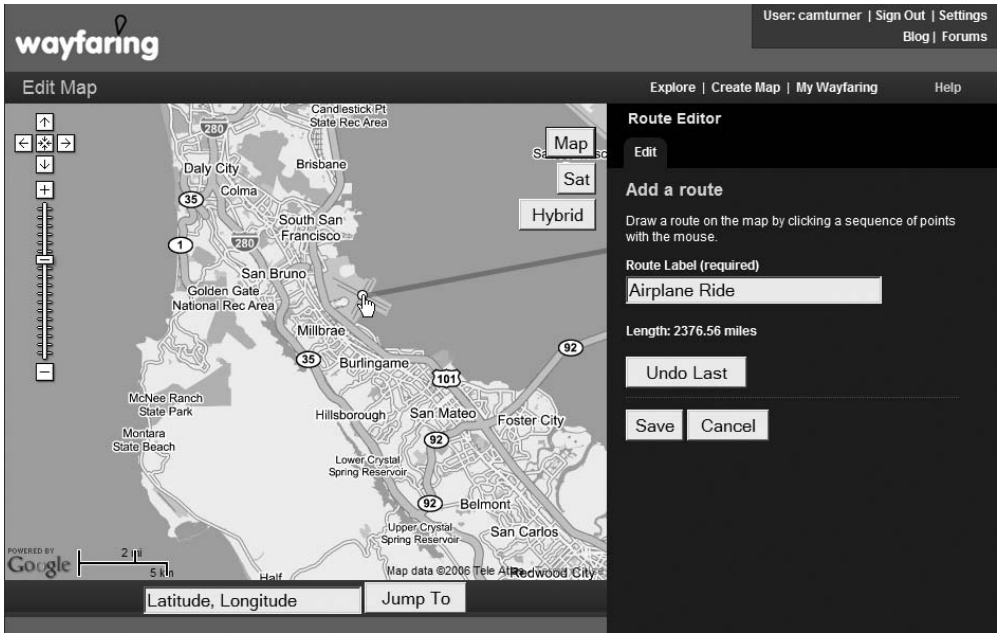


Figure 1-4. *Your flight landing at San Francisco International Airport*

Adding the Destination Point

Now that you're in San Francisco, let's figure out how to get to the Googleplex directly. Click Add a Waypoint. Your destination is Google, so the new point is called *The Googleplex*. Use the address box feature to jump directly to 1600 Amphitheatre Parkway, Mountain View, California, 94043. Wayfaring is able to determine latitude and longitude from an address via a process called *geocoding*, which you'll see a lot more of in Chapter 4.

To confirm you're in the right place, click the Sat button on the top-right corner of the map to switch it over to satellite mode. You should see something resembling Figure 1-5.

Figure 1-5. *The Googleplex*

Adding a Driving Route

Next, let's figure out how far the drive is. Routes don't really have a starting and ending point in Wayfaring from a visual point of view, so you can start your route from the Googleplex and work your way backward. Switch back into Map mode or Hybrid mode so you can see the roads more clearly. From the Map Editor screen, select Add a Route and click the point you just added. Use 10 to 20 dots to carefully trace the trip from Mountain View back up the Bayshore Freeway (U.S. Highway 101) to the airport. You'll end up with about 23 miles of driving, as shown in Figure 1-6.

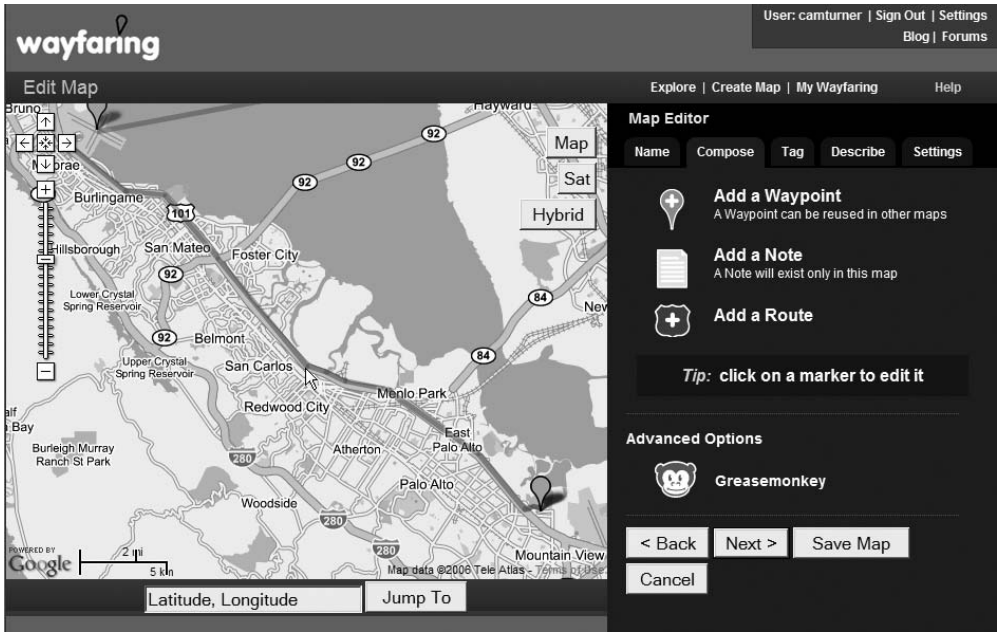


Figure 1-6. *The drive down the Bayshore Freeway to the Googleplex*

That's it. You can use the same principles to make an annotated map of your vacation or calculate how far you're going to travel; and best of all, it's a snap to share it. To see this map live, visit `http://www.wayfaring.com/maps/show/17131`.

Got Rails?

Of course, since this *is* a programming book, you're probably eager to dig into the code and make something really unique. Wayfaring may be nice, but it doesn't give you the flexibility of a programmatic approach. Looking forward to more programmatic interaction with the API, let's discuss Ruby on Rails, the server-side framework of choice.

As we mentioned at the beginning of this chapter, this book presumes you have some experience with Ruby and Rails already—at least enough to get a basic Rails application up and running. In particular, you should

- Have Ruby (1.8.4 or later) and Rails (1.1.5 or later) installed on your development machine.
- Know how to use the `rails` command to create an application skeleton and be comfortable with the model/view/controller layout of a Rails application.
- Know how to start a WEBrick, Mongrel, or other server to view your pages in a browser.
- Have a local database server. We use MySQL throughout this book, but you should be able to adapt to the DB engine of your choice.

- Have a development environment that you're comfortable with—for example, TextMate, RadRails, or Vim.

The following are things we *don't* expect you to know (or have) coming into this book:

- You don't need to know the details of the `prototype.js` JavaScript library. A lot of developers are confused by Prototype, probably due to the dearth of documentation. We've made a conscious decision in this book to not rely heavily on `prototype.js`, although we will be utilizing it from time to time.
- You don't need to know RJS. RJS is a Rails template type (like RHTML) and is a convenient way to build rich functionality on web pages with a minimum of JavaScript coding. Since the Google Maps API is implemented in JavaScript, we'll rely primarily on hand-coded JavaScript functions for this book.
- You don't need to be an expert in JavaScript programming. Yes, the Google Maps API is implemented in JavaScript, but you'll be ramping up on JavaScript techniques and principles as you go.
- You don't necessarily need to have a production web space for your Rails application. Of course, you'll want a production server so you can expose your killer app to the world, but you can learn everything in this book using your local development server.

JUST GETTING STARTED WITH RUBY AND RAILS?

If you're just getting started with Ruby and Rails and decided mapping applications are a fun way to learn, you'll probably want some additional resources to help you get up to speed. Ruby on Rails has a steeper learning curve than PHP or ColdFusion, so you will benefit from these resources to get you started out right:

- *Beginning Ruby: From Novice to Professional* by Peter Cooper (`http://www.apress.com/book/bookDisplay.html?bID=10244`). We recognize that many developers' first exposure to Ruby is through the Rails web framework. If this is the case for you, I highly recommend learning more about the wonderful Ruby language before diving headfirst into Rails.
- *Beginning Ruby on Rails: From Novice to Professional* by Cloves Carneiro Jr. and Jeffrey Allan Hardy (`http://www.apress.com/book/bookDisplay.html?bID=10124`).

What's Next?

We hope you're eager to learn how to build your own maps-based applications from the ground up using Rails. By the end of Part 1 of this book, you'll have the skills to do everything you've just done on Wayfaring (except the polylines and distances, which are covered in Chapter 10) using JavaScript and XHTML. By the book's conclusion, you'll have learned most of the concepts needed to build your own Wayfaring clone.

So what exactly is to come? This book is divided into three parts and two appendixes. Part 1 goes through Chapter 4 and deals with the basics that a hobbyist would need to get started.

You'll make a map, add some custom pins, and geocode a set of data using freely available services. Part 2 (Chapters 5 through 8) gets into more map development topics, such as building a usable interface, dealing with extremely large groups of points, and finding sources of raw information you may need to make your professional map ideas a reality. Part 3 (Chapters 9 through 11) dives into advanced topics: building custom map overlays, such as your own info window and tool tip; creating your own map tiles and projections; using the spherical equations necessary to calculate surface areas on the earth; and building your own geocoder from scratch. Finally, one appendix provides a reference guide to the Google Maps version 2 API, and another points to a few places where you can find neat data for extending the examples here and to inspire your own projects.

CHAPTER 2

Getting Started

In this chapter, you'll learn how to create your first Google map project, plot some markers, and add a bit of interactivity. Because JavaScript plays such a central role in controlling the maps, you'll also start to pick up a few essentials about that language along the way.

In this chapter, we will step through the following:

- Setting up your Rails application
- Getting off the ground with a basic map and a Google Maps API key
- Separating the map application's JavaScript functions and XHTML
- Unloading finished maps to help browsers free their memory
- Creating map markers and responding to clicks on them with an information pop-up

On JavaScript, Helpers, and Plug-ins

Rails' baked-in integration with Prototype and Scriptaculous via helper libraries is a godsend for many common JavaScript and Ajax use cases. Since the helpers cover so many common cases, many Rails developers never dig deep into JavaScript itself. If that's the case for you, get ready to learn some JavaScript. The Google Maps API is 100% JavaScript, and you need to get familiar with JavaScript to fully leverage all the API's mapping goodness.

Likewise, there are some Google Maps-related plug-ins listed on RubyForge, which will provide Google Maps-specific helpers. The two Google Maps-related plug-ins listed on RubyForge are Cartographer (`http://cartographer.rubyforge.org/`) and YM4R (`http://rubyforge.org/projects/ym4r/`). Cartographer has been around longer (according to RubyForge), but YM4R is more mature and full-featured. YM4R also seems to be more up-to-date and actively maintained.

YM4R provides an attractive way to begin working with Google Maps on Rails with minimal upfront JavaScript programming. If you choose to use it, you'll find excellent documentation at `http://thepochisuperstarmegashow.com/ProjectsDoc/ym4r_gm-doc/`.

The approach taken in this book is to interact with the Google Maps with JavaScript, the API's native tongue. Doing so has several advantages:

- You get access to the full range of capabilities of the API. You are not relying on any intermediary between your code and Google Maps.
- There are fewer layers to debug when something goes wrong with your code.
- It's easier to evolve your application to take advantage of updates to the Google Maps API. Google has demonstrated that the Maps API will undergo frequent updates. If you're working against the raw API, you are not dependent upon a third-party to upgrade a plug-in in order to take advantage of new capabilities.

Ultimately, we believe that you will be well served by learning the Google Maps API in its native JavaScript. If you later decide to utilize a plug-in like YM4R, you will possess a fundamental understanding of the JavaScript code that the plug-in produces. You will also be able to extend the plug-in or intermingle its output with raw JavaScript if necessary.

Creating Your Rails Application

You need a Rails application to hold your Google Maps code. Let's create an application now from the command line by changing into the directory in which your Rails application will live, and executing the `rails` command, passing it the name of our application. We'll call our application *maps*, so type `rails maps` from the command line. Now create a `chap_two` controller with the command `ruby script/generate controller chap_two`. You'll use this controller to get started. Note that in this chapter, you are only working with HTML and JavaScript; you don't technically need any Ruby code yet. In fact, you could run all the examples in this chapter as plain HTML files under Apache or IIS (Internet Information Services), if you are so inclined. But this is a Rails book; so why not use the Rails environment from the very start?

Before going on, make sure to note the port that your application will be running on. For example, if you're developing with RadRails, the WEBrick server generator assigns a port to each newly created server. If you already have applications running on ports 3000 and 3001, your new application will be on port 3002. You'll need to know the port to get your Google Maps key, which is the next step. From now on, we will use `http://localhost:3000` as our development URL; if yours is different, substitute your actual URL as you go.

The First Map

In this section, you'll obtain a Google Maps API key and then begin experimenting with it by retrieving Google's starter map code.

Keying Up

Before you start a Google Maps web application, you need to sign up for a Google Maps API key. To obtain your key, you must accept the Google Maps API Terms of Use. At the time of this writing, a few of the important terms included not stealing Google's imagery, obscuring the Google logo, or holding Google responsible for its software. Additionally, you're prevented from creating maps that invade privacy or facilitate illegal activities. Terms of use can change, so make sure you check them over before you plan your Google Maps–based application.

Google will issue as many keys as you need, but separate domains (or different ports on the same domain) *must* apply for a separate key. Your first key will be used in your local development environment; so enter **http://localhost:3000** as your domain. Visit `http://www.google.com/apis/maps/signup.html` (Figure 2-1) and submit the form to get your key. Throughout this book, nearly all of the examples will require you to include this key in the JavaScript `<script>` element for the Google Maps API, as you're about to see in Listing 2-1.

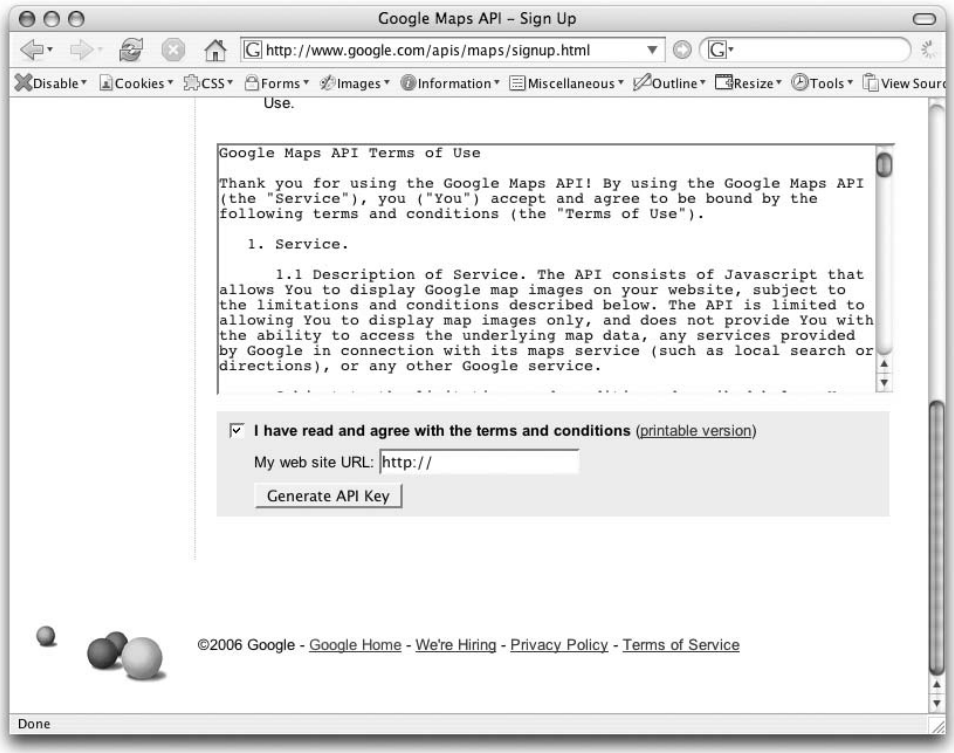


Figure 2-1. *To sign up for an API key, check the box and enter the URL of your webspace.*

■ **Note** Why a key? It's common practice for API providers to require a key, so they can monitor usage, identify the most popular projects, and note potential terms of service violations. Google is not the only one that makes you authenticate to use an API. Del.icio.us, Amazon, and others all provide services with APIs that require you to first obtain a key.

When you sign up to receive your key, Google will provide you with some very basic starter code to help you get a map up and running as quickly as possible. We'll begin by dissecting and working with this starter code so you can gain a basic understanding of what's happening.

If you start off using Google's sample, your key is already embedded in the JavaScript. Alternatively, you can—as with all listings—grab the source code from this book's web site at `http://googlemapsbook.com` and insert your own key by hand.

Either way, save the code to a file called `map.rhtml` in the directory `views/chap_two/`. Your key is that long string of characters following `key=` (Our key is `ABQIAAAA33EjxkLYsh9SEveh_MphphQP1y➥R2bHJW2Brl_bW_lOKXsyt8cxTKO5Zz-UKoJ6IepTlZRxN8nfTRgw`). Listing 2-1 shows the contents of your `views/chap_two/map.rhtml` file.

Listing 2-1. *The Google Maps API Starter Code in views/chap_two/map.rhtml*

```
<!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>
        <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
        <title>Google Maps JavaScript API Example</title>
        <script src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAA➥
33EjxkLYsh9SEveh_MphphQP1yR2bHJW2Brl_bW_lOKXsyt8cxTKO5Zz-UKoJ6Ie➥
pTlZRxN8nfTRgw" type="text/javascript"></script>
        <script type="text/javascript">

        //<![CDATA[

        function load() {
            if (GBrowserIsCompatible()) {
                var map = new GMap2(document.getElementById("map"));
                map.setCenter(new GLatLng(37.4419, -122.1419), 13);
            }
        }

        //]]>
        </script>
    </head>

    <body onload="load()" onunload="GUnload()">
        <div id="map" style="width: 500px; height: 300px"></div>
    </body>
</html>
```