A

TALE OF TWO CITIES.

BY

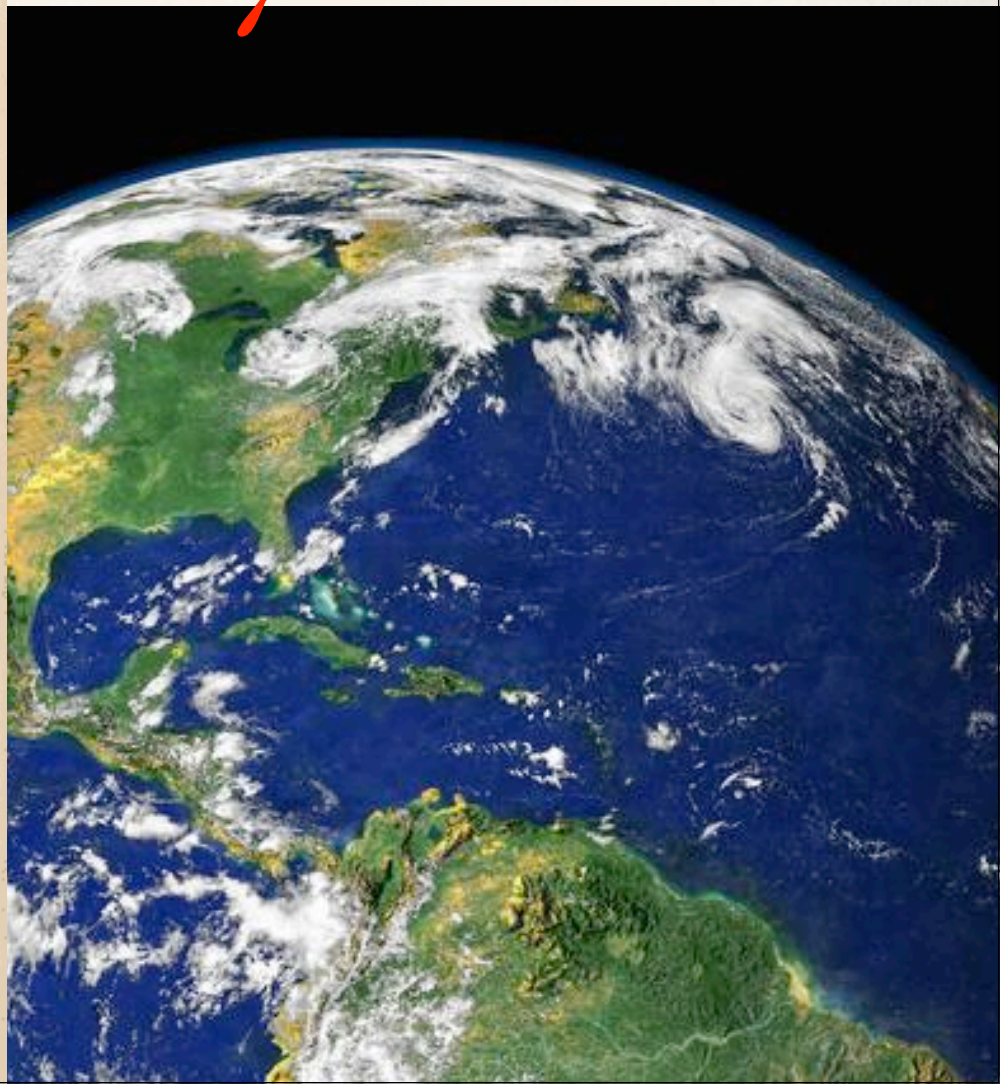
CHARLES DICKENS.

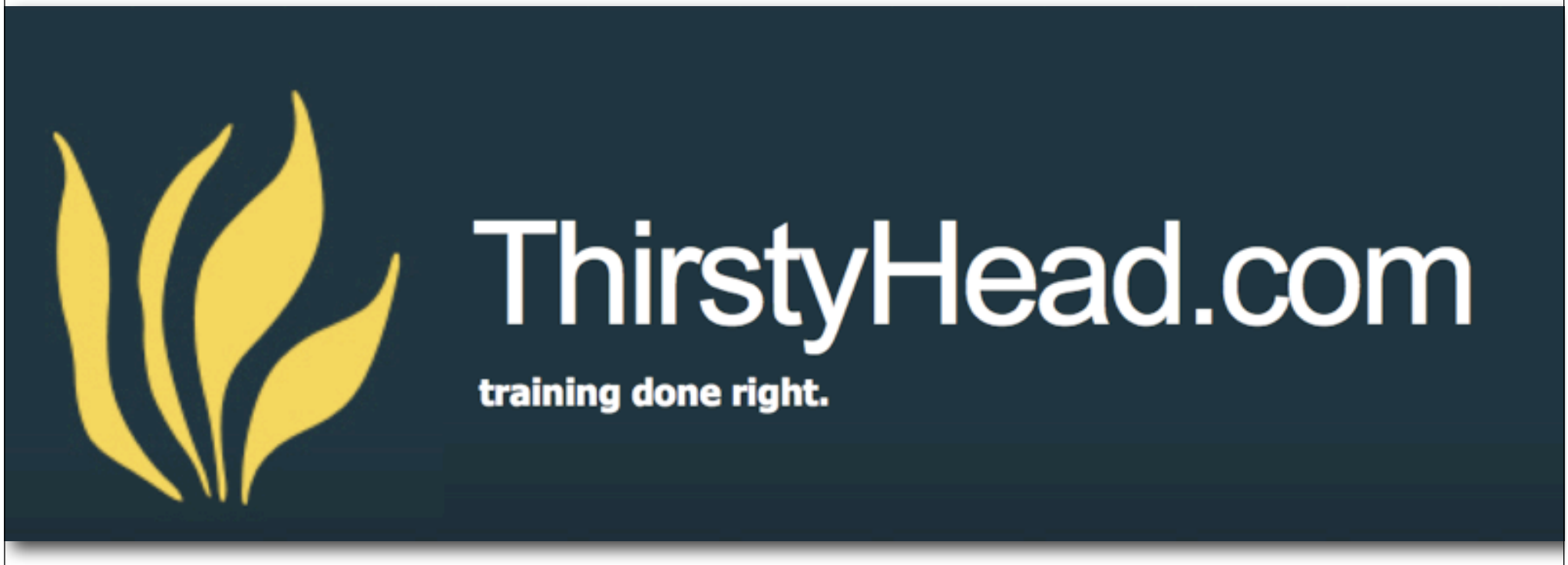
WITH ILLUSTRATIONS BY H. K. BROWNE.

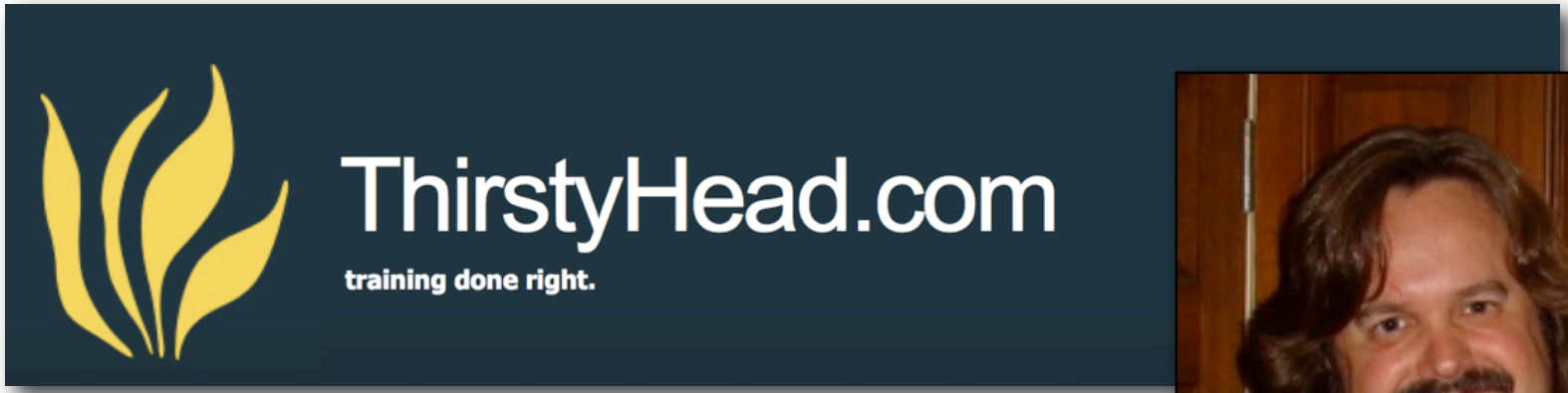
LONDON:
CHAPMAN AND HALL, 193, PICCADILLY;
AND AT THE OFFICE OF ALL THE YEAR ROUND,
11, WELLINGTON STREET NORTH.
MDCCCLIX.

Slippy Maps

(...and Scott Davis)




ThirstyHead.com
training done right.



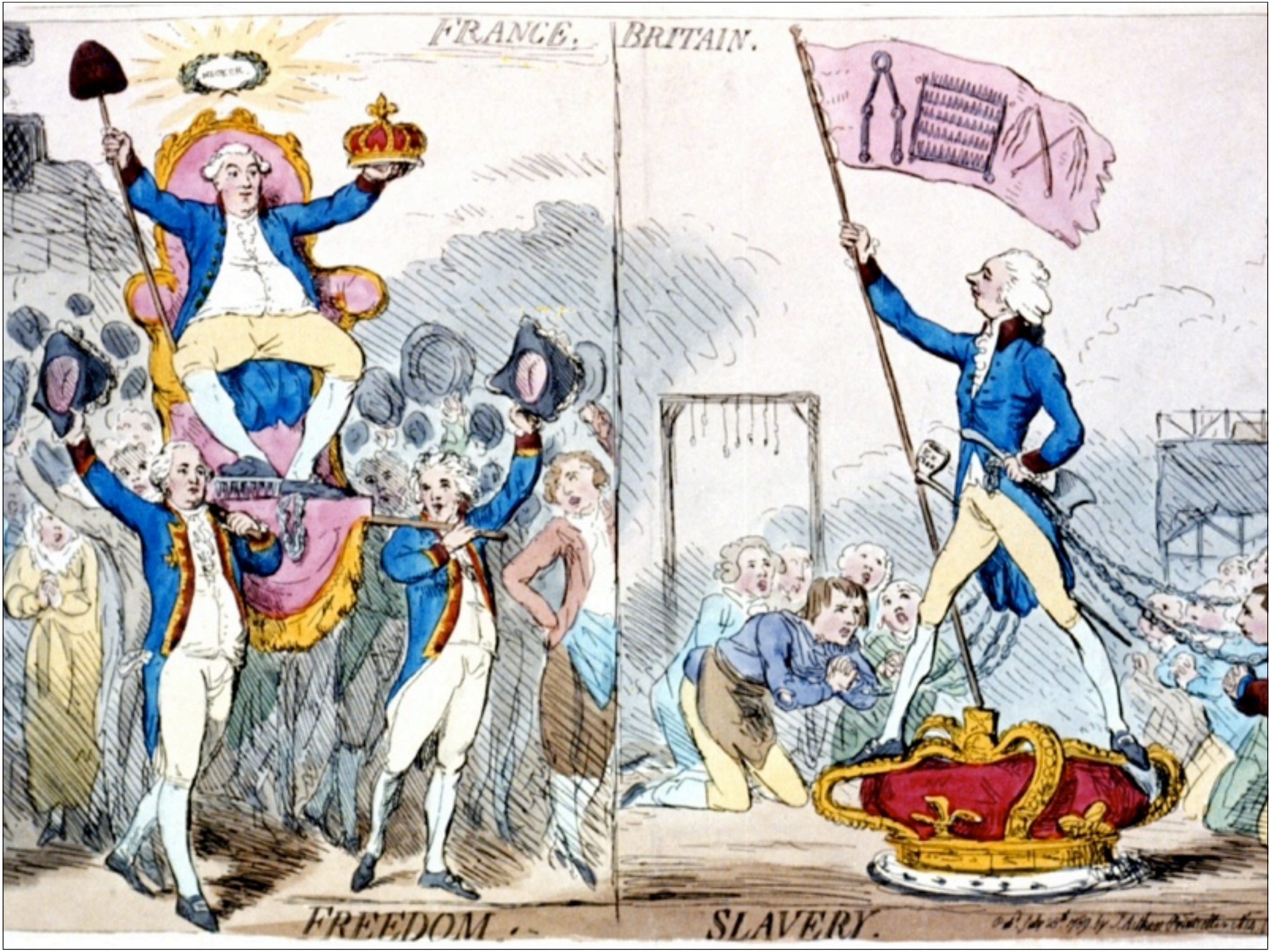
Scott Davis

IBM

developerWorks > Java technology

Mastering Grails

Practically Groovy


FRANCE.
BRITAIN.
FREEDOM.
SLAVERY.

It was the best of times,

it was the worst of times...

Ajax: A New Approach to Web Applications

Jesse James Garrett is President and a founder of Adaptive Path. He is the author of the widely-referenced book The Elements of User Experience. Jesse's other essays include The Nine Pillars of Successful Web Teams and Six Design Lessons From the Apple Store.

To get essays like this one delivered directly to your inbox,

by Jesse James Garrett

February 18, 2005

If anything about current interaction design can be called "glamorous," it's creating Web applications. After all, when was the last time you heard someone rave about the interaction design of a product that wasn't on the Web? (Okay, besides the iPod.) All the cool, innovative new projects are online.

Despite this, Web interaction designers can't help but feel a little envious of our colleagues who create desktop software. Desktop applications have a richness and responsiveness that has seemed out of reach on the Web. The same simplicity that enabled the Web's rapid proliferation also creates a gap between the experiences we can provide and the experiences users can get from a desktop application.

That gap is closing. Take a look at Google Suggest. Watch the way the suggested terms update as you type, almost instantly. Now look at Google Maps. Zoom in. Use your cursor to grab the map and scroll around a bit. Again, everything happens almost instantly, with no waiting for pages to reload.

Google Suggest and Google Maps are two examples of a new approach to web applications that we at Adaptive Path have been calling Ajax. The name is shorthand for Asynchronous JavaScript + XML, and it represents a fundamental shift in what's possible on the Web.

Google Suggest and Google Maps are two examples
of a new approach to web applications that we at Adaptive Path have been calling Ajax.
The name is shorthand for Asynchronous JavaScript + XML, and it represents a
fundamental shift in what's possible on the Web.

Static Maps

vs.

Slippy Maps

MAPQUEST

Explore the world with the #1 consumer travel site!*

FIND IT — Let's Go!

Check out:
- Interactive Atlas (Maps)
- TripQuest (Driving directions)
- TravelPlan USA (Plan a trip with Mobil Travel Guide)

GET MORE INFO — Let's Go!

Get more information on:
- Free membership • MoveQuest
- Shop for merchandise • Contest
- Share maps & tips • Survey

LEARN ABOUT — Let's Go!

Learn about:
- Advertising • Clients
- Mapping services • Press info

It's a moving experience!

Find your dream home:
- Browse daily MLS Listings
- Zoom in to street maps
- Interactive Moving Calendar
- School reports and more!

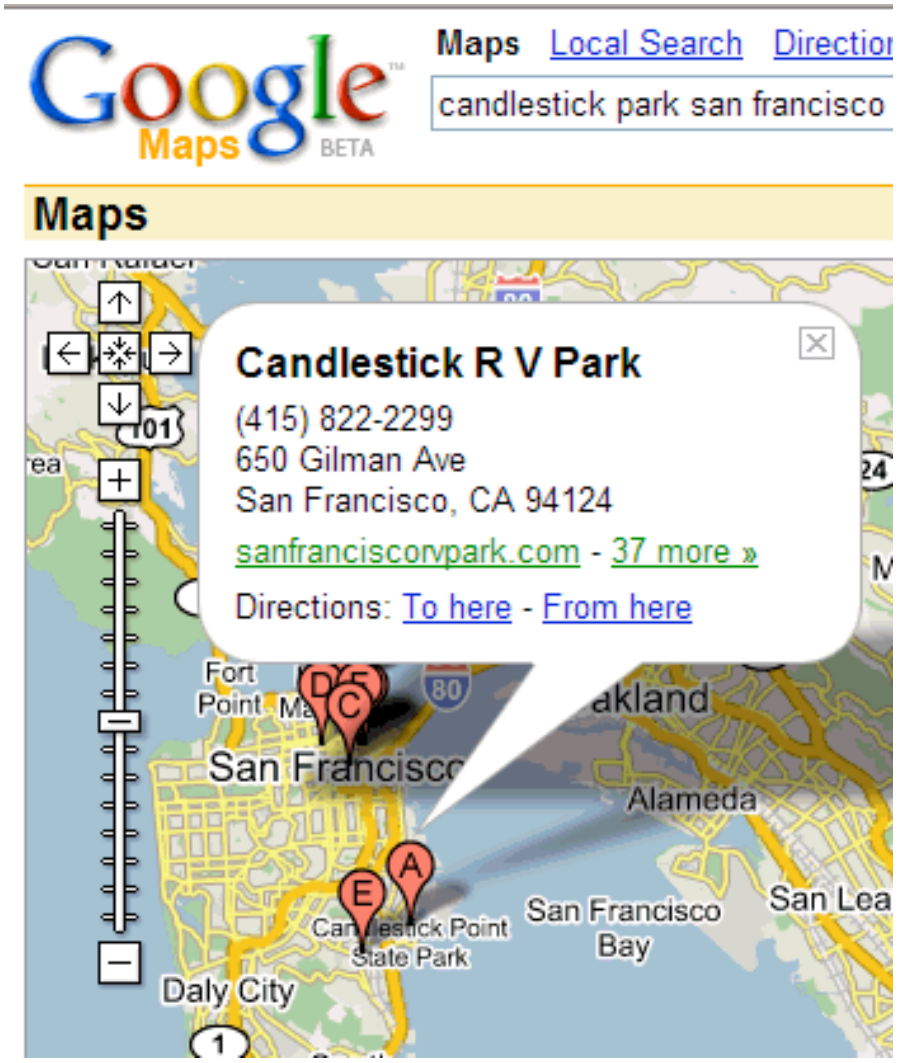
Feel at home in a new neighborhood:
- LOCAL information.
- Schools
- Housing Costs
- Crime Areas and more!

Make a move: click HERE to visit MoveQuest.

[Help] [About] [Feedback] [FAQ] [What's New] [Home]

---

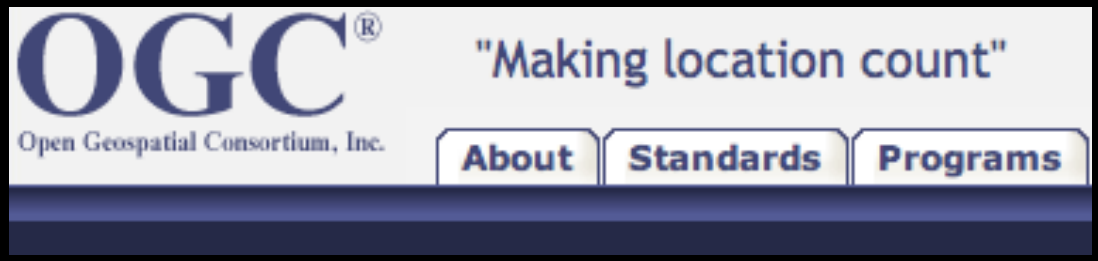
* Media Metrix (PC Meter), July 1997


Google Maps BETA
Maps
Local Search
candlestick park san francisco
Maps
Candlestick R V Park
(415) 822-2299
650 Gilman Ave
San Francisco, CA 94124
sanfranciscorvpark.com - 37 more »
Directions: To here - From here
Fort Point
San Francisco
Alameda
Candlestick Point State Park
San Francisco Bay
Daly City
101
80
24

Who is the OGC?

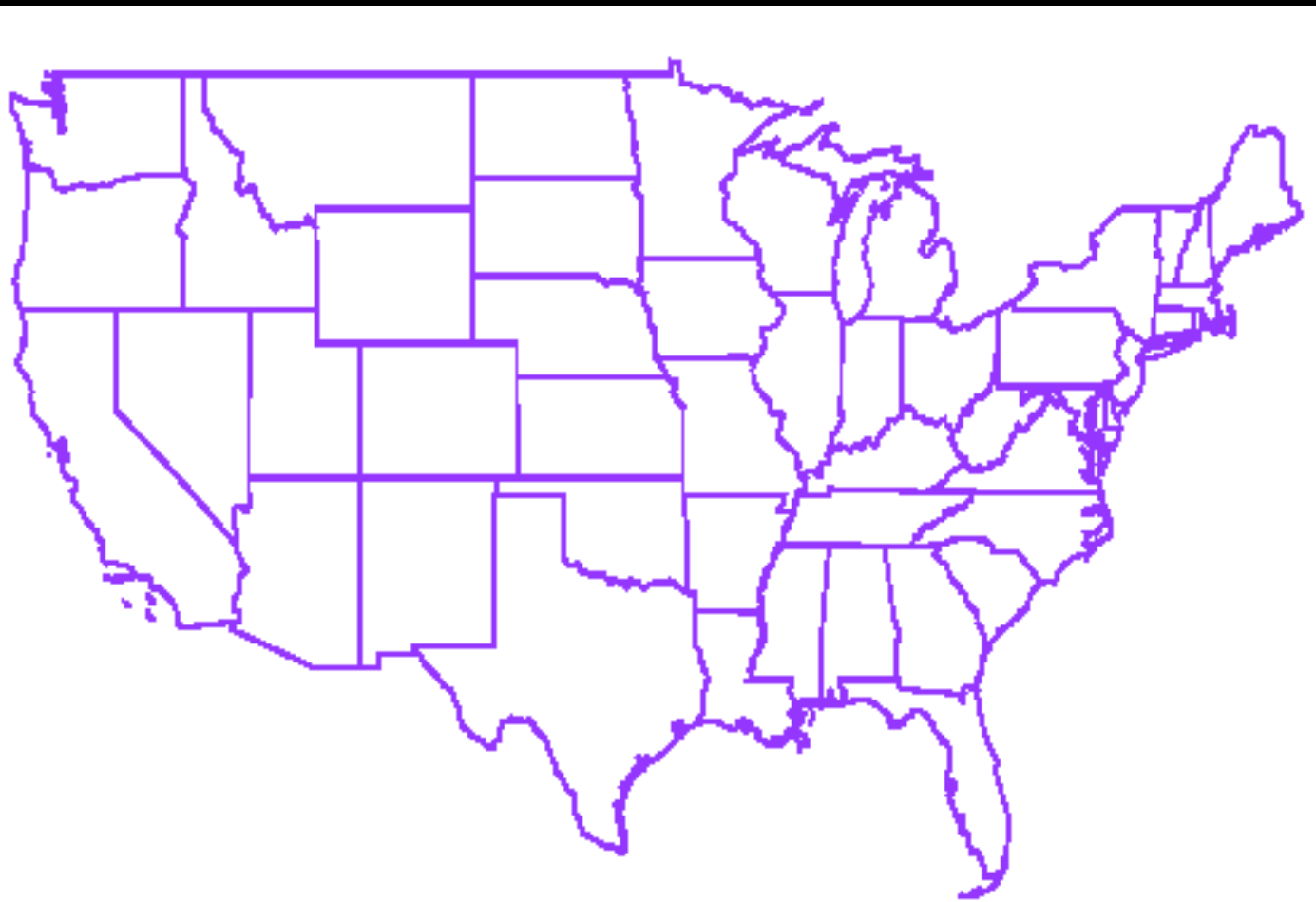
- The Open Geospatial Consortium
  - http://www.opengis.org

The Open Geospatial Consortium, Inc. (OGC) is a non-profit, international, voluntary consensus standards organization that is leading the development of standards for geospatial and location based services.



Web Mapping Service (WMS)

http://localhost:8888/geoserver/wms?
VERSION=1.1.1&
REQUEST=GetMap&
SRS=EPSG:4326&
BBOX=-126,20,-66,52&
WIDTH=500&
HEIGHT=500&
LAYERS=us_states_poly&
STYLES=&
FORMAT=image/png&
BGCOLOR=0xffffff&
TRANSPARENT=FALSE&
EXCEPTIONS=application/vnd.ogc.se_inimage



Composition

- The WMS spec predates Google Maps
  - The intent is map composition, not slippy maps
  - Each request is painstakingly assembled, lovingly rendered, and then thrown away

Decisions, Decisions

Slippy Map	WMS
Built for speed	Built for customization
Many tiles	One big tile
Fixed zoom levels	Variable zoom levels
Fixed map layers	Variable map layers
Fixed image format	Variable image format

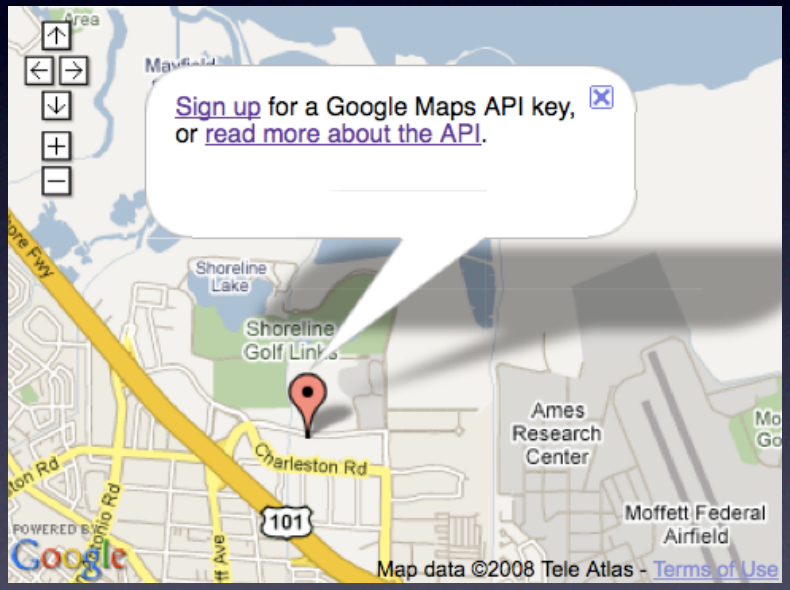
About WMS-C

The WMS Tiling Client Recommendation, was the result of a set of discussions at the FOSS4G conference in 2006. The recommendation describes a way to constrain WMS requests to a predefined grid, so that clients can request data that has been pre-rendered or rendered on the fly and the cached. Delivering cached map imagery can reduce image load times from the client's point of view by as much as one or even two orders of magnitude.

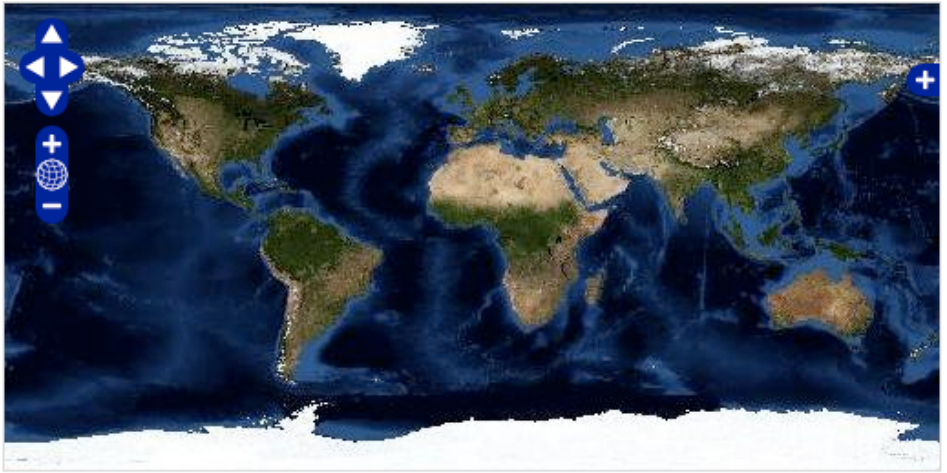
Although some users of WMS have attempted to use naive HTTP proxies to solve the problems of WMS caching, caching HTTP proxies alone don't take full advantage of prior knowledge of the WMS protocol, and don't account for the various idiosyncrasies that WMS clients can present.

Rendering map imagery on the fly for every WMS request typically requires profound hardware resources to scale well, creating a barrier to entry for organizations wishing to offer WMS access to their data. We hope that creating simple-to-deploy solutions for caching map imagery will help to push forward the publication of geographic data via WMS servers, which will in turn make more and richer maps available to more people. At MetaCarta Labs, we intend to further this goal by developing software that implements WMS caching servers and clients, such as OpenLayers and TileCache.

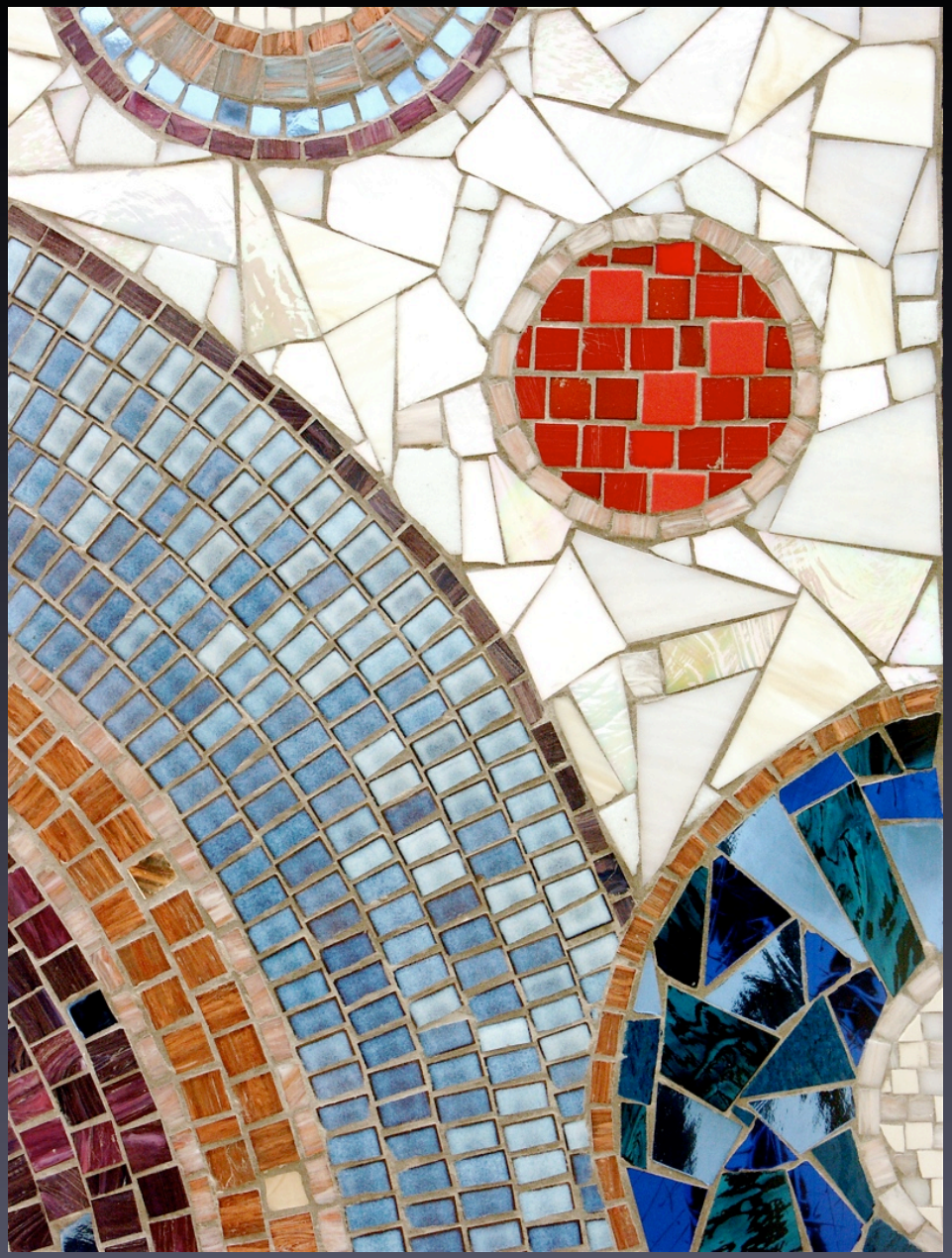
Google Code



OpenLayers



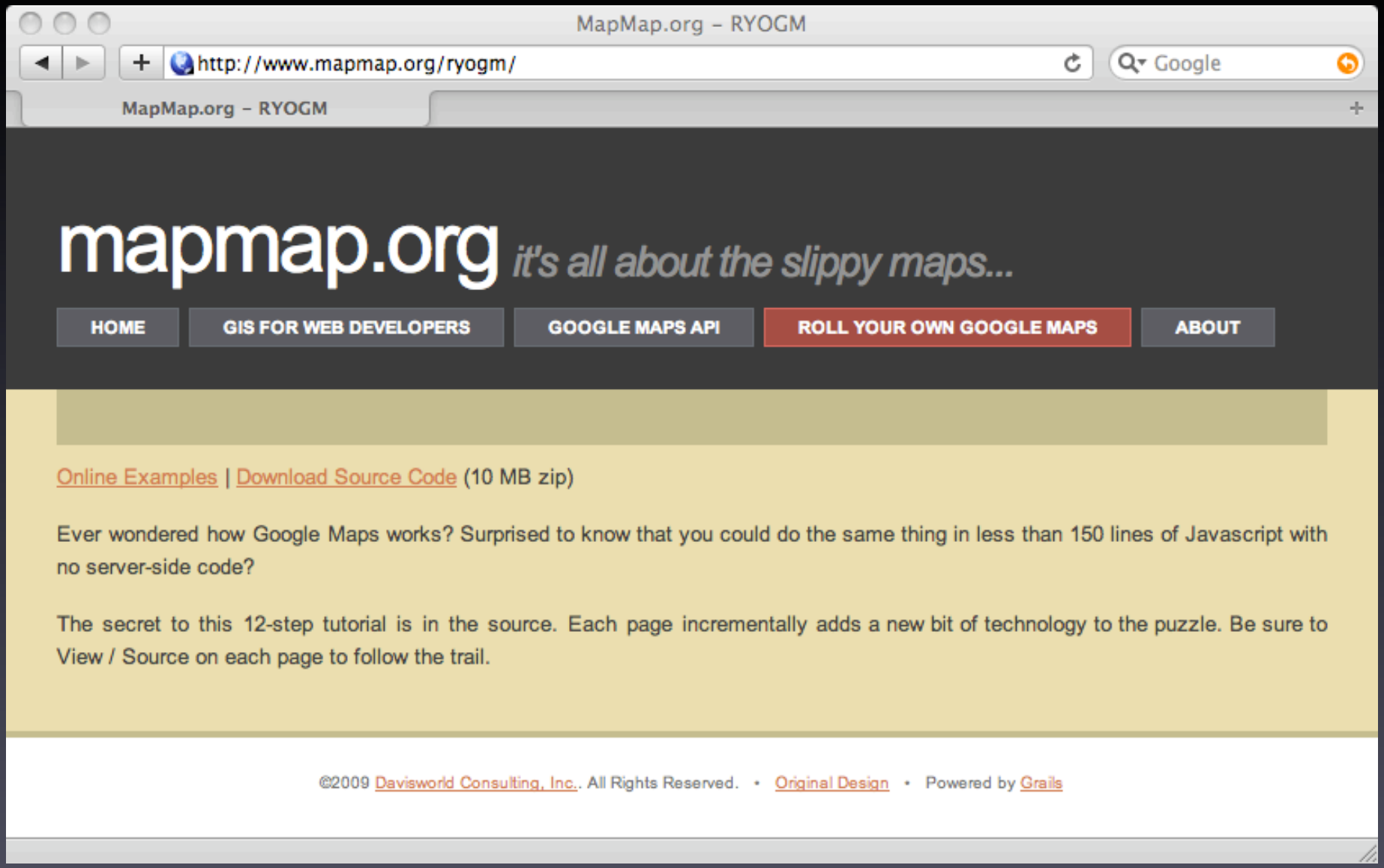
Put an open map widget in any web page!
Double-click to zoom in, and drag to pan. Hold down the shift key and drag to zoom to a particular region.

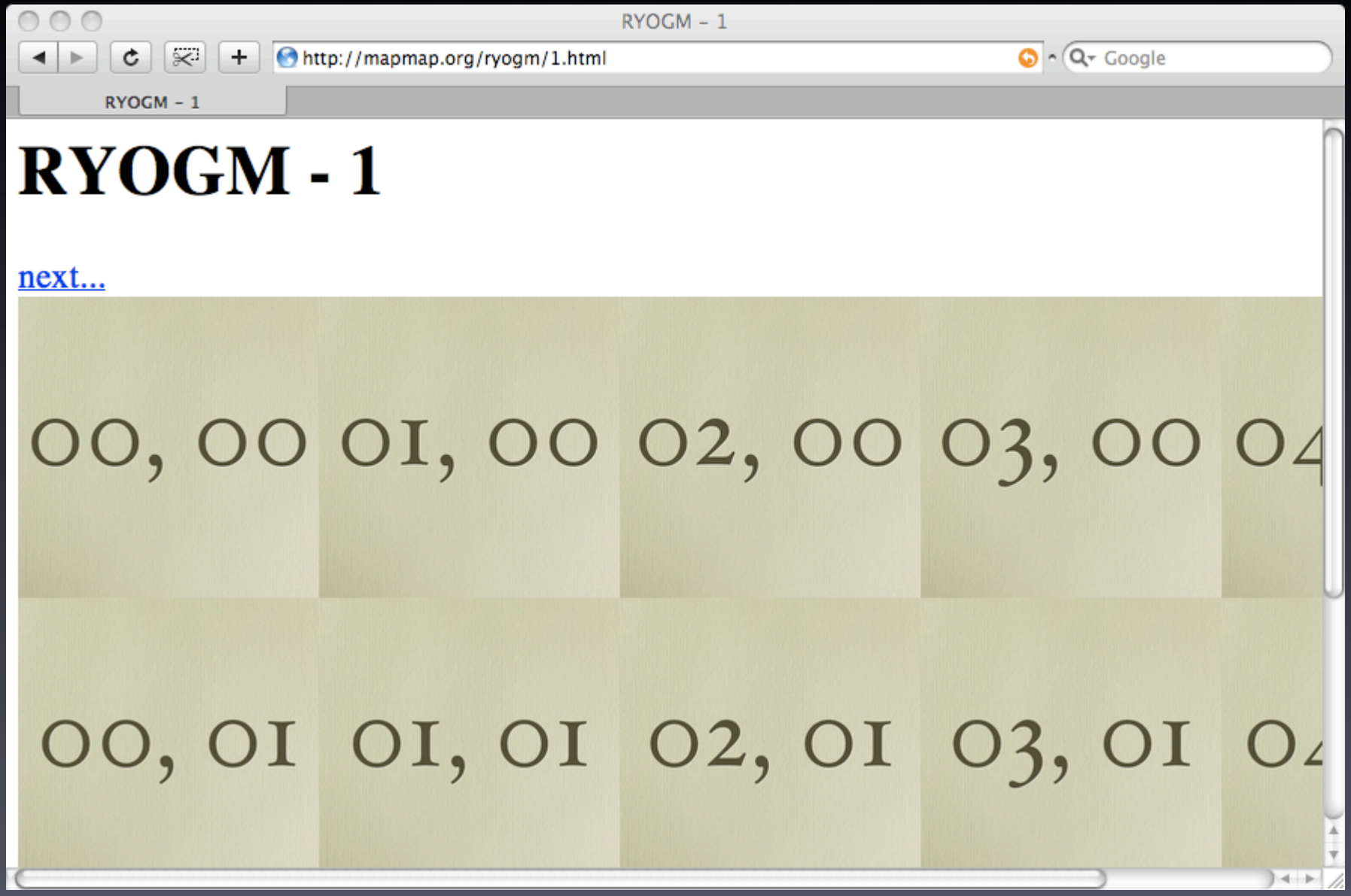


Tiled

Tessellated

Mosaicked


MapMap.org – RYOGM
http://www.mapmap.org/ryogm/
Google
MapMap.org – RYOGM
mapmap.org
it's all about the slippy maps...
HOME
GIS FOR WEB DEVELOPERS
GOOGLE MAPS API
ROLL YOUR OWN GOOGLE MAPS
ABOUT
Online Examples | Download Source Code (10 MB zip)
Ever wondered how Google Maps works? Surprised to know that you could do the same thing in less than 150 lines of Javascript with no server-side code?
The secret to this 12-step tutorial is in the source. Each page incrementally adds a new bit of technology to the puzzle. Be sure to View / Source on each page to follow the trail.
©2009 Davisworld Consulting, Inc.. All Rights Reserved. • Original Design • Powered by Grails


RYOGM – 1
http://mapmap.org/ryogm/1.html
Google
RYOGM – 1
RYOGM - 1
next...
00, 00
01, 00
02, 00
03, 00
00, 01
01, 01
02, 01
03, 01

Source of http://localhost:8080/ryogm/1.html

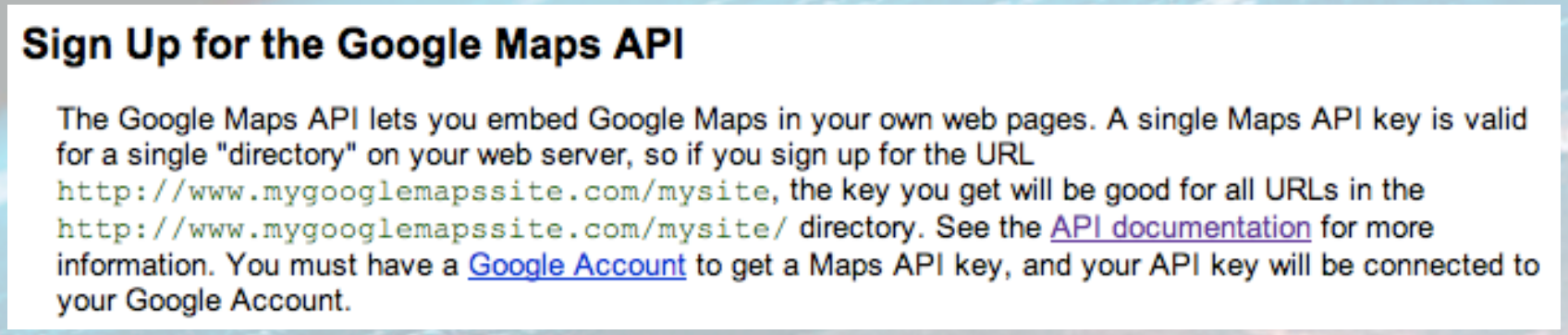
```
<html>
<head>
   <title>RYOGM - 1</title>
   <!-- lay out the table -->
   <!-- Don't freak out -- we'll dynamically build this in just a moment. -->
</head>

<body>
<h1>RYOGM - 1</h1>
<a href="2.html">next...</a>

<table border="0" cellpadding="0" cellspacing="0">
   <tr>
      <td><img src="images/test/00-00.png"/></td>
      <td><img src="images/test/01-00.png"/></td>
      <td><img src="images/test/02-00.png"/></td>
      <td><img src="images/test/03-00.png"/></td>
      <td><img src="images/test/04-00.png"/></td>
   </tr>
   <tr>
      <td><img src="images/test/00-01.png"/></td>
      <td><img src="images/test/01-01.png"/></td>
      <td><img src="images/test/02-01.png"/></td>
      <td><img src="images/test/03-01.png"/></td>
      <td><img src="images/test/04-01.png"/></td>
   </tr>
   <tr>
      <td><img src="images/test/00-02.png"/></td>
      <td><img src="images/test/01-02.png"/></td>
      <td><img src="images/test/02-02.png"/></td>
      <td><img src="images/test/03-02.png"/></td>
      <td><img src="images/test/04-02.png"/></td>
   </tr>
```

- Google Maps is one of many free APIs
- Others:
  - http://developer.yahoo.com/maps/
    - Offers Ajax, Flash, and GeoRSS
  - http://dev.live.com/virtualearth/
    - Microsoft's Offering

- To begin, we need to sign up for a free Google Maps key
  - http://www.google.com/apis/maps/signup.html



Sign Up for the Google Maps API

The Google Maps API lets you embed Google Maps in your own web pages. A single Maps API key is valid for a single "directory" on your web server, so if you sign up for the URL `http://www.mygooglemapssite.com/mysite`, the key you get will be good for all URLs in the `http://www.mygooglemapssite.com/mysite/` directory. See the API documentation for more information. You must have a Google Account to get a Maps API key, and your API key will be connected to your Google Account.

Google Maps API

Google Code Home >Google Maps API >Google Maps API Signup

Google Maps API
Sign Up for an API Key
Create a KML Sitemap

API Documentation

Concepts and Examples
Reference

Resources

API Help
API Terms of Use
API Blog
API Discussion Group

Google Maps for Enterprise

Includes enterprise licensing and support

Thank you for signing up for a Google Maps API key!

Your key is:

```
ABQIAAAAU7YnM-urF7eE9yC49kPN7hSSBtlNIjNHi5GgKsdinkxZzfIUuRQOhFR
```

This key is good for all URLs in this directory:

```
http://localhost:8080/mapping/
```

Here is an example web page to get you started on your way to mapping glory:

```
<!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
    <title>Google Maps JavaScript API Example</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIA
      type="text/javascript"></script>
    <script type="text/javascript">
```

Example Page

Here is an example web page to get you started on your way to mapping glory:

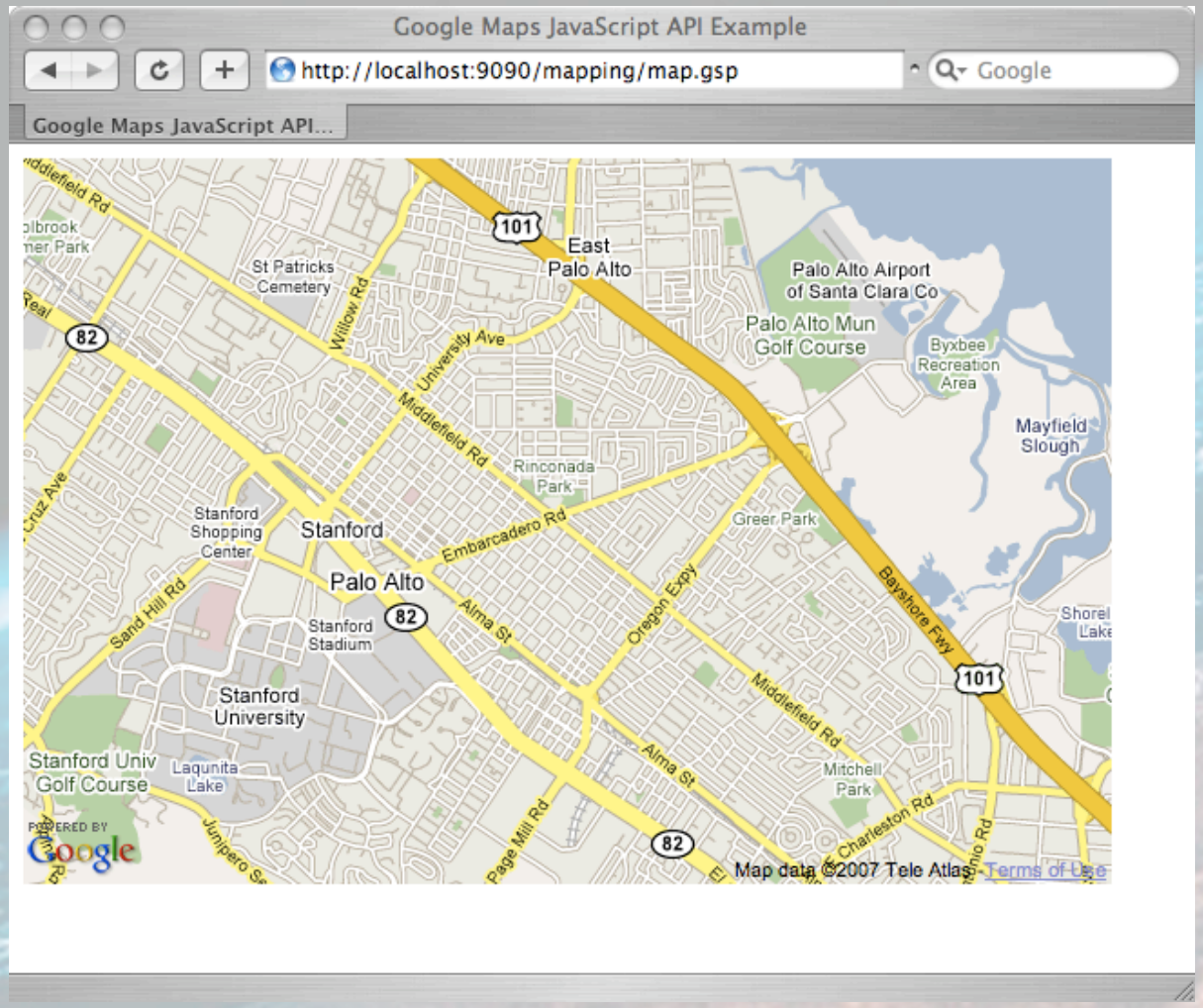
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAU7YnM-ur
      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>
```

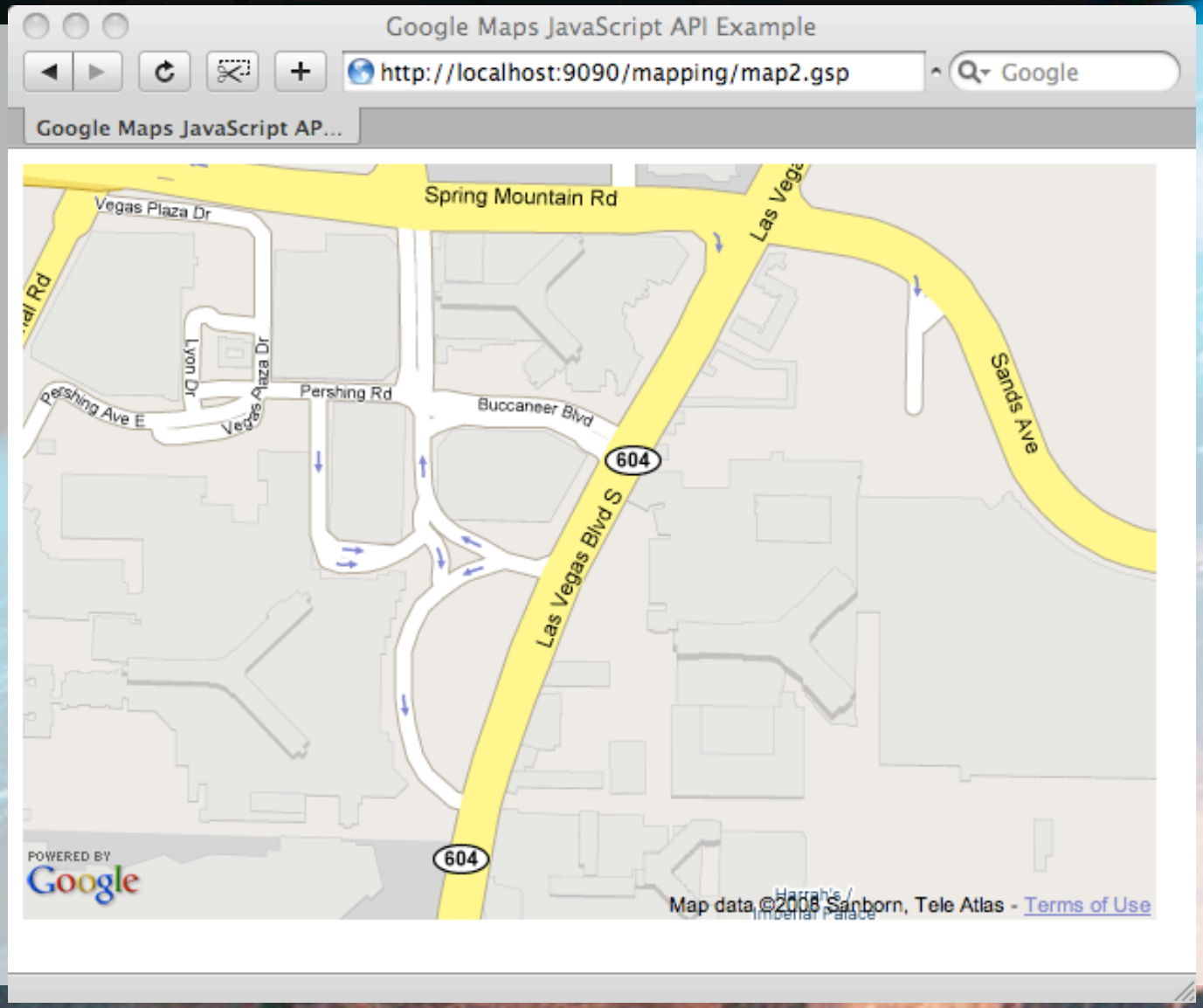
Here is our first map



- Adjusting our lat/lon and zoom level:
  - Zoom 0 == World
  - Zoom 20 == Street Level

```
function load() {
  if (GBrowserIsCompatible()) {
    var map = new GMap2(document.getElementById("map"));
    map.setCenter(new GLatLng(36.122487,-115.171181), 16);
  }
}
```

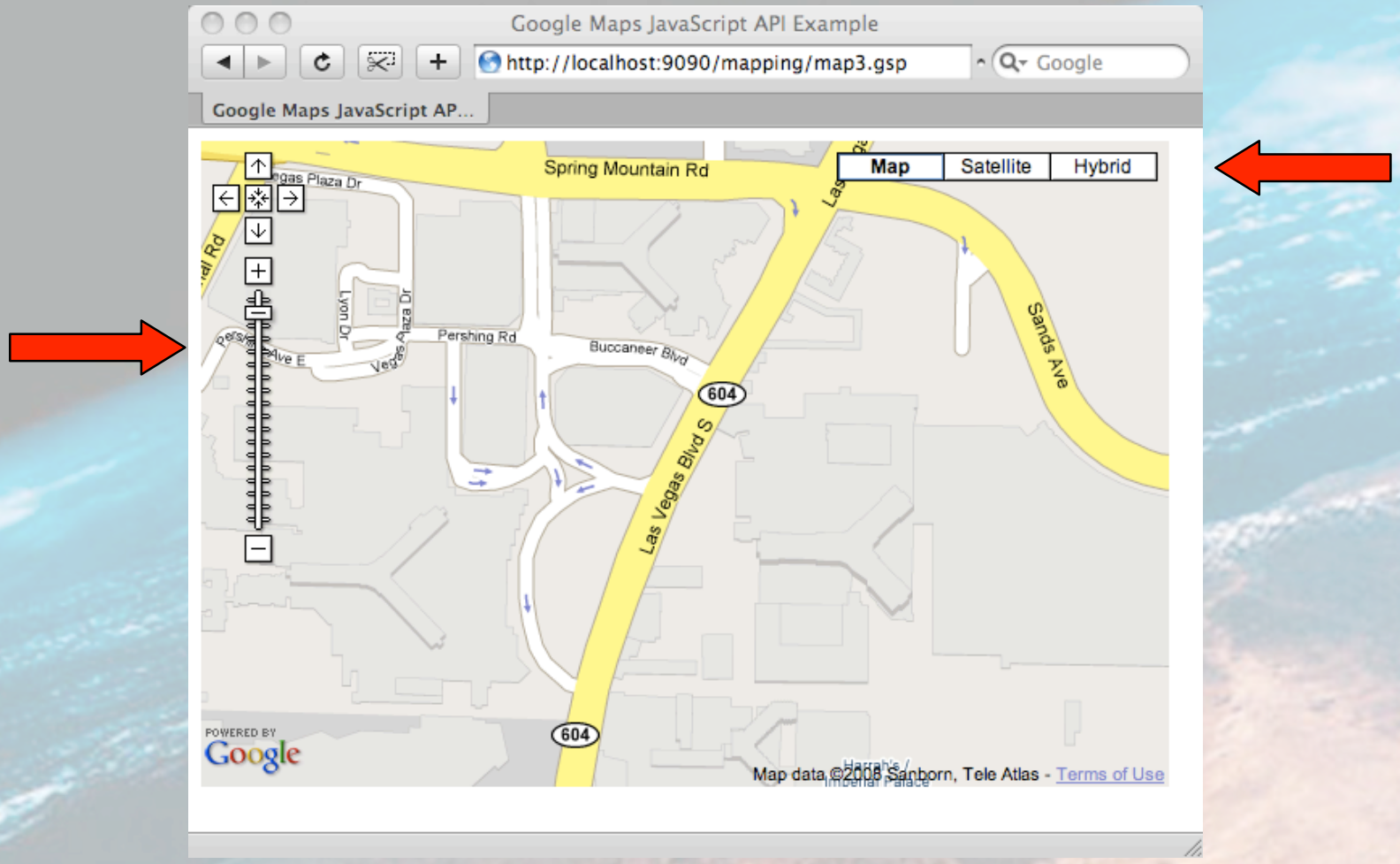
Map, Round 2


Google Maps JavaScript API Example
http://localhost:9090/mapping/map2.gsp
Google
Google Maps JavaScript AP...
Spring Mountain Rd
Vegas Plaza Dr
Lyon Dr
Vegas Plaza Dr
Pershing Ave E
Pershing Rd
Buccaneer Blvd
Las Vegas Blvd S
604
Sands Ave
604
POWERED BY
Google
Map data ©2008 Sanborn, Tele Atlas - Terms of Use

- Let's add in some controls:

```
function load() {
  if (GBrowserIsCompatible()) {
    var map = new GMap2(document.getElementById("map"));
    map.setCenter(new GLatLng(36.122487,-115.171181), 16);
    map.addControl(new GLargeMapControl())
    map.addControl(new GMapTypeControl())
  }
}
```

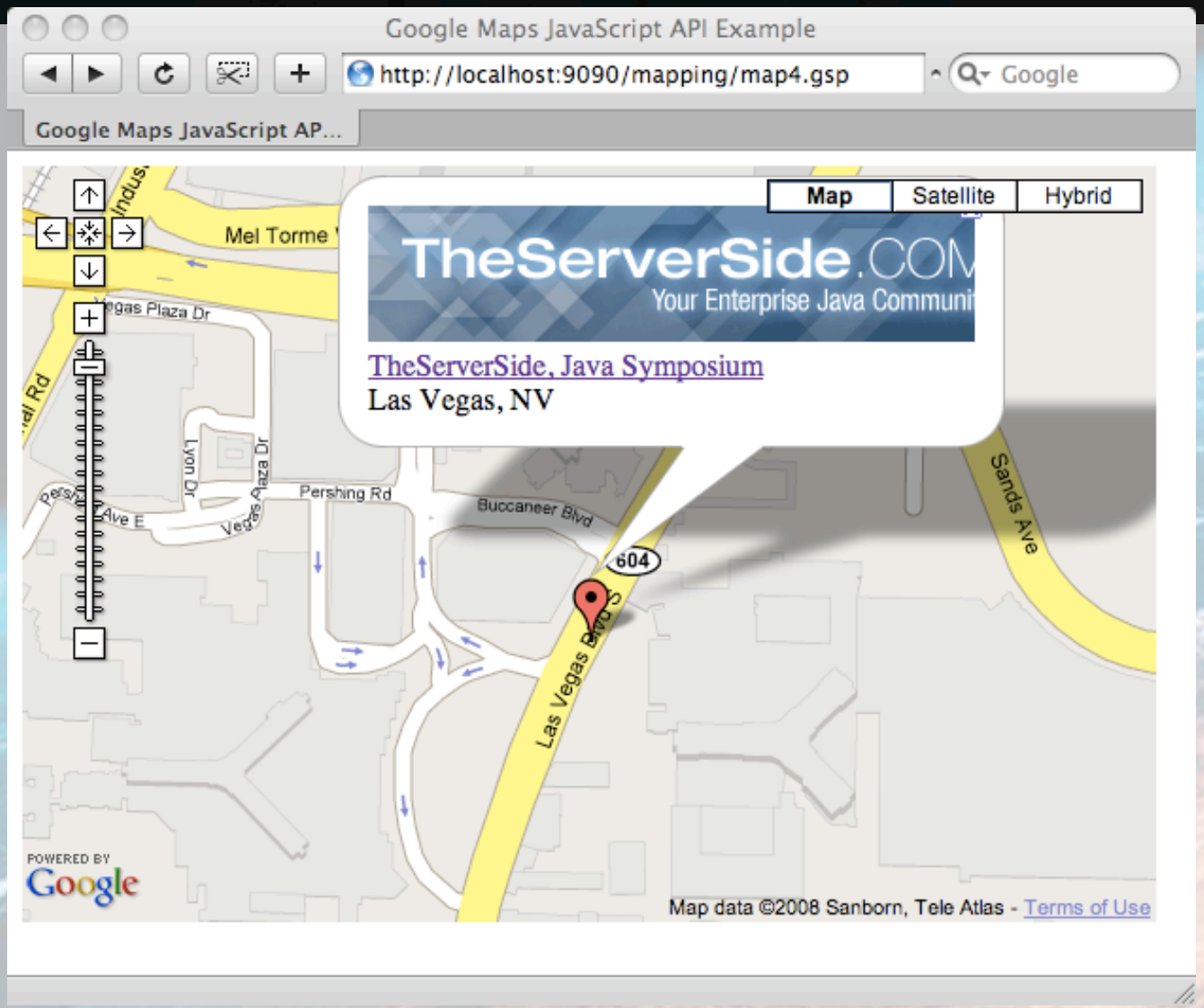
Map, Round 3

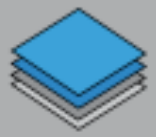

Google Maps JavaScript API Example
http://localhost:9090/mapping/map3.gsp
Google
Google Maps JavaScript AP...
Map
Satellite
Hybrid
Spring Mountain Rd
Pershing Rd
Buccaneer Blvd
Las Vegas Blvd S
Sands Ave
Lyon Dr
Vegas Plaza Dr
604
604
POWERED BY
Google
Map data ©2008 Sanborn, Tele Atlas - Terms of Use

- Finally, let's add a marker and an info window:

```
var logoUrl = "http://www.theserverside.com/tt/skin/
images/header_logotype.gif"
var bcDesc = "<img src='" + logoUrl + "' width='321'
height='72'/>"
bcDesc += "<br />"
bcDesc += "<a href='http://javasymposium.techtarget.com/
lasvegas/index.html'>"
bcDesc += "TheServerSide, Java Symposium"
bcDesc += "</a><br />"
bcDesc += "Las Vegas, NV"
bcMarker.openInfoWindowHtml(bcDesc)
```

Map, Round 4

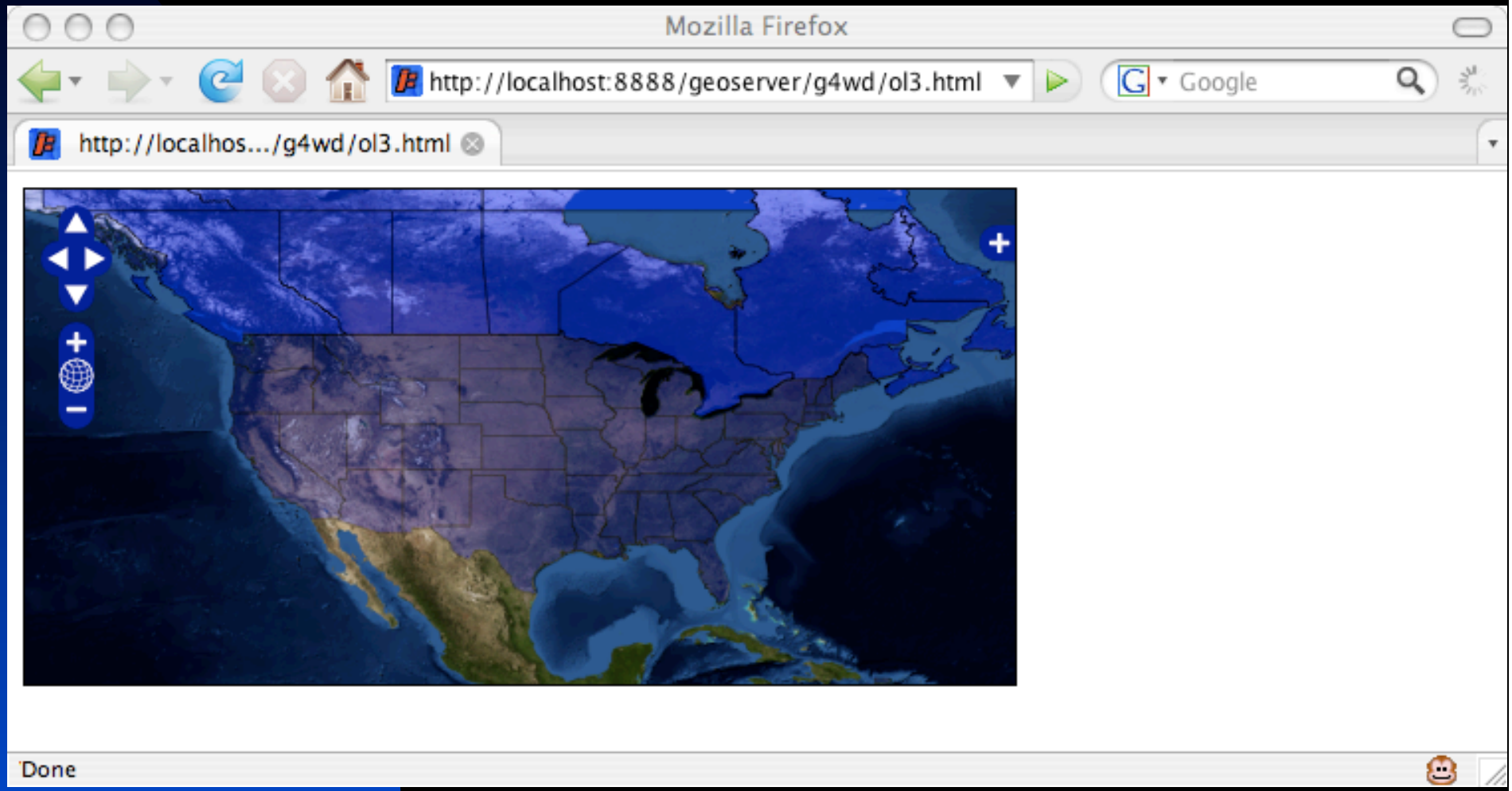

Google Maps JavaScript API Example
http://localhost:9090/mapping/map4.gsp
Google
Google Maps JavaScript AP...
Map
Satellite
Hybrid
TheServerSide.COM
Your Enterprise Java Communi
TheServerSide, Java Symposium
Las Vegas, NV
Mel Torme
Vegas Plaza Dr
Lyon Dr
Vegas Plaza Dr
Pershing Rd
Buccaneer Blvd
Sands Ave
604
Las Vegas Blvd S
POWERED BY
Google
Map data ©2008 Sanborn, Tele Atlas - Terms of Use



OpenLayers

- Pure JavaScript web client
  - http://www.openlayers.org
- In addition to supporting OGC layers
  - Google Maps
  - Yahoo Maps
  - MS Virtual Earth (Local Live)
  - WorldWind

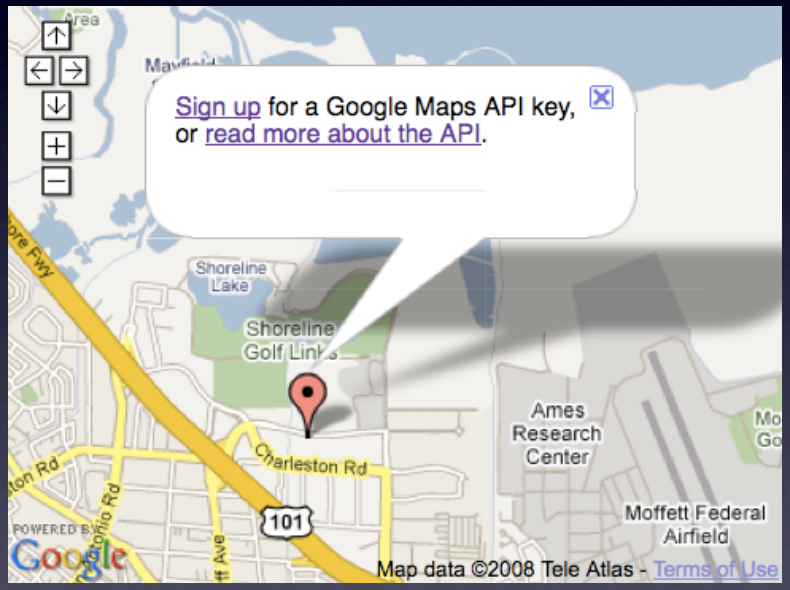
One Map…


Mozilla Firefox
http://localhost:8888/geoserver/g4wd/ol3.html
Google
http://localhos.../g4wd/ol3.html
Done

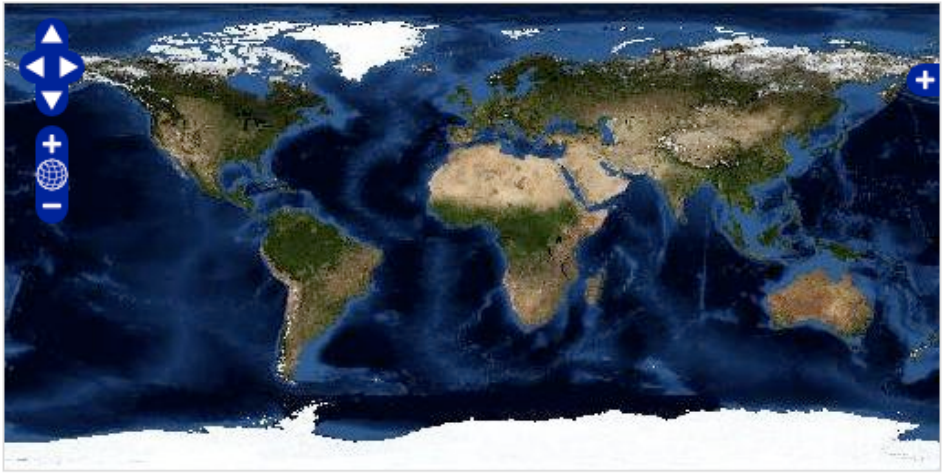
…One File

```
10      <script src="http://www.openlayers.org/api/OpenLayers.js"></script>
11      <script type="text/javascript">
12          //NOTE: geographic center of the US
13          var lon = -98.583333;
14          var lat = 39.833333;
15          var zoom = 3;
16          var map, us, canada, blueMarble;
17
18          function init(){
19              map = new OpenLayers.Map( $('map') );
20              blueMarble = new OpenLayers.Layer.WMS( "Blue Marble",
    "http://wms.jpl.nasa.gov/wms.cgi?", {layers: 'BMNG', format: 'image/png'},
    {isBaseLayer:true});
21              map.addLayer(blueMarble);
22
23              us = new OpenLayers.Layer.WMS( "US", "http://localhost:8888/geoserver/wms?",
    {layers: 'g4wd:st99_d00', format: 'image/png', transparent: true}, {isBaseLayer:false,
    opacity:0.5} );
24              map.addLayer(us);
25
26              canada = new OpenLayers.Layer.WMS( "Canada",
    "http://localhost:8888/geoserver/wms?", {layers: 'g4wd:prov_ab_p_geo83_e', format:
    'image/png', transparent: true}, {isBaseLayer:false, opacity:1.0} );
27              map.addLayer(canada);
28
29              map.setCenter(new OpenLayers.LonLat(lon, lat), zoom);
30              map.addControl( new OpenLayers.Control.LayerSwitcher() );
31          }
32      </script>
```

Google Code



OpenLayers



Put an open map widget in any web page!
Double-click to zoom in, and drag to pan. Hold down the shift key and drag to zoom to a particular region.

Two Dimensional Maps

vs.

Three Dimensional Maps

Google Launches Free 3D Mapping and Search Product

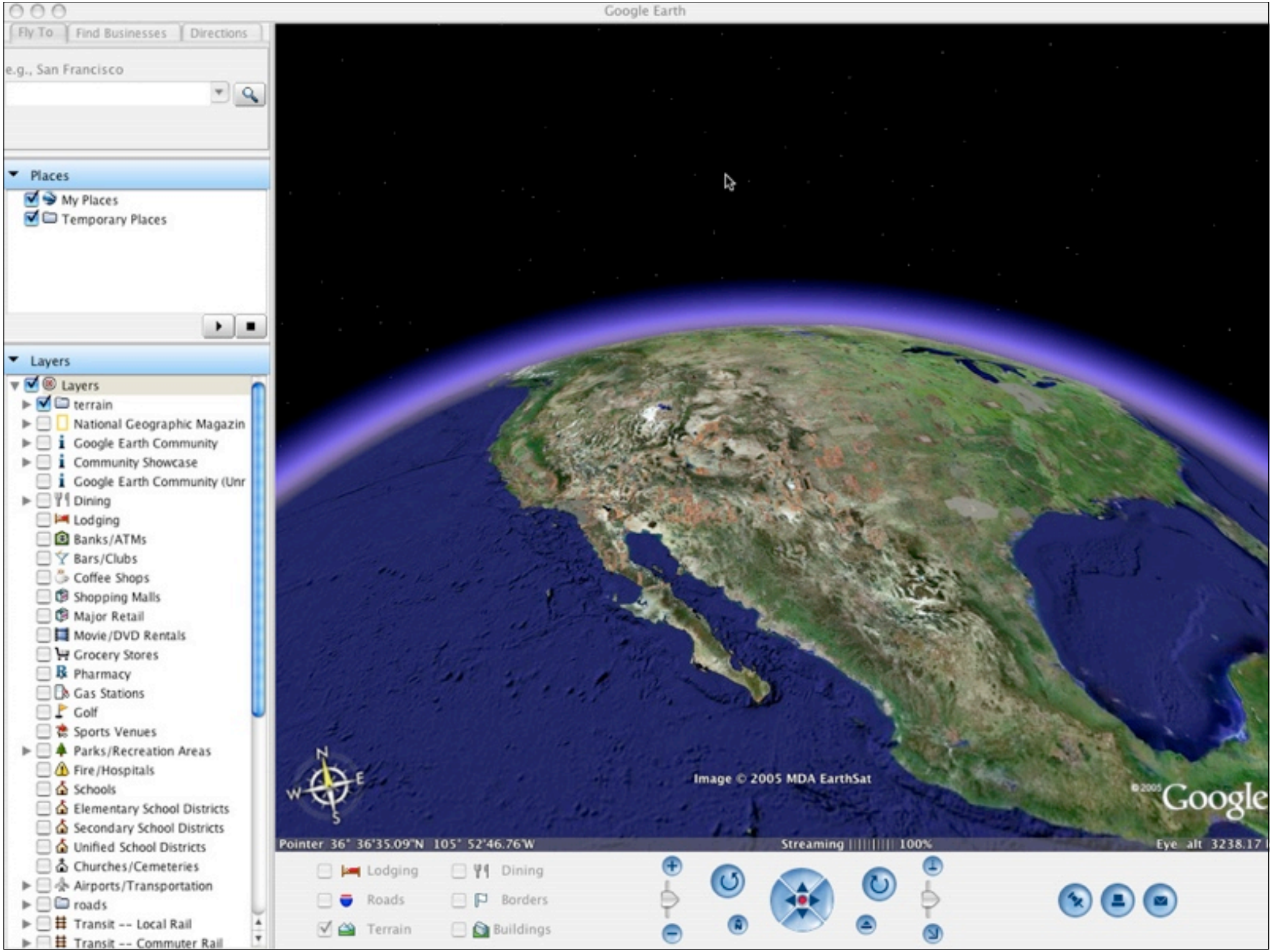
Google Earth Offers New Local Search Experience

MOUNTAIN VIEW, Calif. - June 28, 2005 - Google Inc. (NASDAQ: GOOG) today announced the launch of Google Earth, Google's new satellite imagery-based mapping product that combines 3D buildings and terrain with mapping capability and Google search. Based on Keyhole technology, Google Earth enables users to fly from space to street level views to find geographic information and explore places around the world.

Key features of Google Earth include:

- Free software download available at http://earth.google.com
- 3D buildings in major cities across the United States
- 3D terrain showing mountains, valleys, and canyons around the world
- Integrated Google Local search to find local information such as hotels, restaurants, schools, parks, and transportation
- Fast, dynamic navigation
- Video playback of driving directions
- Tilt, rotate, and activate 3D terrain and buildings for a different perspective on a location
- Easy creation and sharing of annotations among users

"Google Earth utilizes broadband streaming technology and 3D graphics, much like a videogame, enabling users to interactively explore the world-- either their own neighborhood or the far corners of the globe," said John Hanke, general manager, Keyhole, Google Inc. "With many ways to access geographic information, Google provides a very rich local search experience for users worldwide."

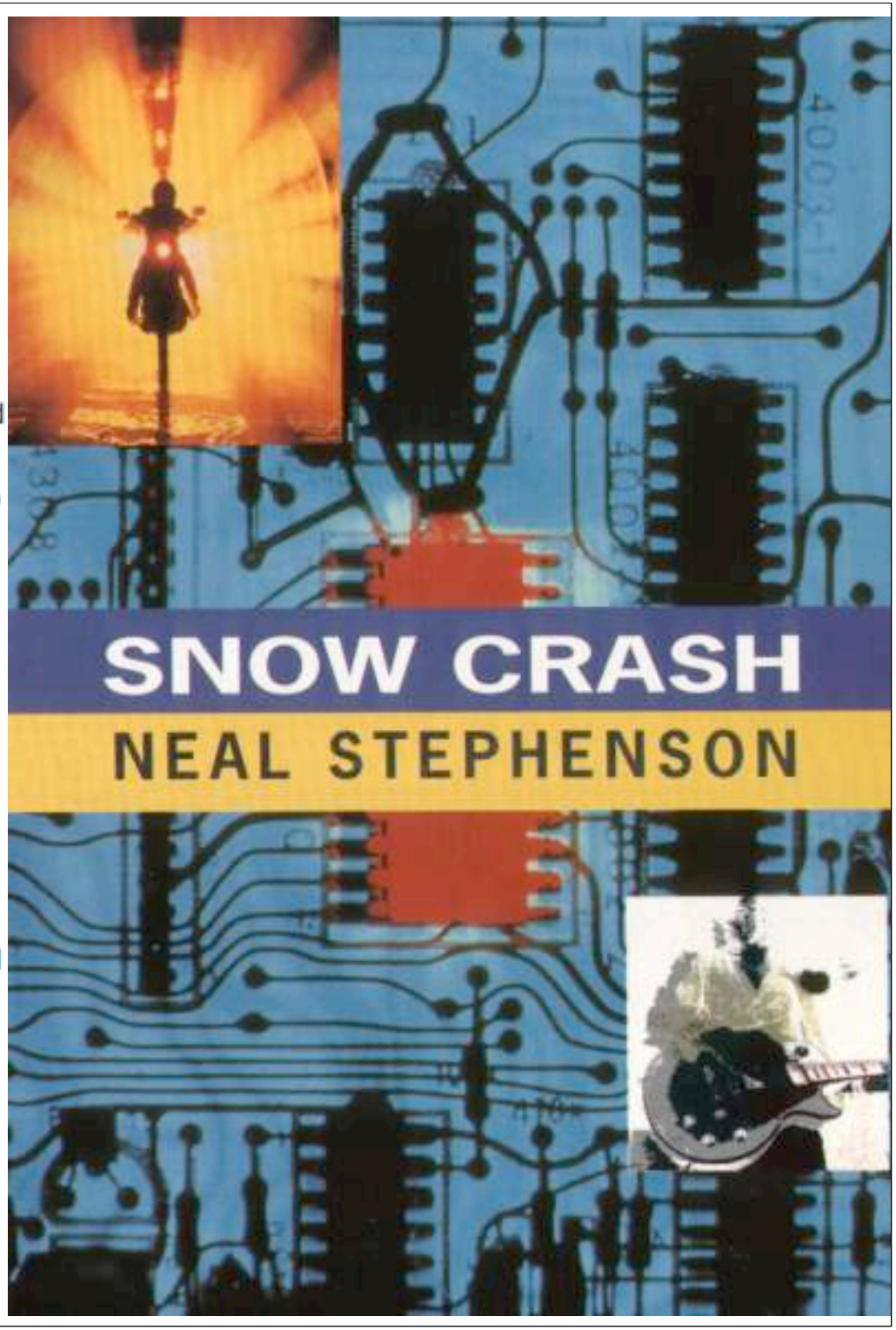

Google Earth
Fly To
Find Businesses
Directions
e.g., San Francisco
Places
My Places
Temporary Places
Layers
Layers
terrain
National Geographic Magazin
Google Earth Community
Community Showcase
Google Earth Community (Unr
Dining
Lodging
Banks/ATMs
Bars/Clubs
Coffee Shops
Shopping Malls
Major Retail
Movie/DVD Rentals
Grocery Stores
Pharmacy
Gas Stations
Golf
Sports Venues
Parks/Recreation Areas
Fire/Hospitals
Schools
Elementary School Districts
Secondary School Districts
Unified School Districts
Churches/Cemeteries
Airports/Transportation
roads
Transit -- Local Rail
Transit -- Commuter Rail
Image © 2005 MDA EarthSat
©2005 Google
Pointer 36° 36'35.09"N 105° 52'46.76"W
Streaming 100%
Eye alt 3238.17
Lodging
Dining
Roads
Borders
Terrain
Buildings

Web User: **Your company Keyhole Inc created Earth Viewer, which later became Google Earth, back in 2001. What inspired you to set about the task of mapping the entire globe?**

John Hanke: Around that time, the first 3D hardware accelerators were being created for consumer PCs, and broadband was starting to roll out in the US, which was a really big deal. We thought now's the time to do this type of visualisation, as it will no longer only be available to the elite. We were just really enamoured of the technology, and approached it as something people would like as much as we did, as we thought it was so cool.

We talked about things it could be used for – planning travel, finding an apartment – and a little bit about something that seemed really far out to us then, which was that users might be able to edit locations with audio or video clips.

Also, a couple of us had read the sci-fi novel *Snow Crash* [by Neal Stephenson], in which there's a software application the lead character, Hiro Protagonist, uses called Earth. That was definitely an inspiration – when were starting, we were all thinking about it - and Earth made the transition from science fiction to reality in a pretty short period of time.



Google Earth is able to show all kinds of images overlaid on the surface of the earth and is also a Web Map Service (WMS) client.

Google Earth supports managing three-dimensional Geospatial data through Keyhole Markup Language (KML).

http://en.wikipedia.org/wiki/Google_earth

Keyhole Markup Language

From Wikipedia, the free encyclopedia

"KML" redirects here. For other uses, see KML (disambiguation).

Keyhole Markup Language (**KML**) is an XML-based language schema for expressing geographic annotation and visualization on existing or future Web-based, two-dimensional maps and three-dimensional Earth browsers. KML was developed for use with Google Earth, which was originally named Keyhole Earth Viewer. It was created by Keyhole, Inc, which was acquired by Google in 2004. The name "Keyhole" is an homage to the KH reconnaissance satellites, the original eye-in-the-sky military reconnaissance system first launched in 1976. KML is an international standard of the Open Geospatial Consortium. Google Earth was the first program able to view and graphically edit KML files, and other projects such as Marble have also started to develop KML support.[1]

Geography Markup Language

From Wikipedia, the free encyclopedia

The **Geography Markup Language** (**GML**) is the XML grammar defined by the Open Geospatial Consortium (OGC) to express geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. Note that the concept of feature in GML is a very general one and includes not only conventional "vector" or discrete objects, but also coverages (see also GMLJP2) and sensor data. The ability to integrate all forms of geographic information is key to the utility of GML.

Data vs Pretty Pixels

- Data services:
  - WFS (Web Feature Service)
- Portrayal services:
  - WMS (Web Mapping Service)

GetCapabilities

- The first thing a program will do is ask the OGC server what data layers it has to offer
  - This is called a GetCapabilities request

```
http://localhost:8888/geoserver/wfs?
service=WFS&
version=1.0.0&
request=GetCapabilities
```

GetCapabilities

- And here is the response:

```
- <FeatureTypeList>
  - <Operations>
      <Query/>
    </Operations>
  - <FeatureType>
      <Name>wfs:co_cities_poly</Name>
      <SRS>EPSG:4326</SRS>
      <LatLongBoundingBox minx="-109.018116" miny="37.00000485324387" maxx="-1
    </FeatureType>
  - <FeatureType>
      <Name>wfs:co_counties_poly</Name>
      <SRS>EPSG:4326</SRS>
      <LatLongBoundingBox minx="-180.0" miny="-90.0" maxx="180.0" maxy="90.0"/>
    </FeatureType>
  - <FeatureType>
      <Name>wfs:co_highways_line</Name>
      <SRS>EPSG:4326</SRS>
      <LatLongBoundingBox minx="-109.06008268030426" miny="36.9936887423584" m
    </FeatureType>
  - <FeatureType>
      <Name>wfs:us_states_poly</Name>
      <SRS>EPSG:4326</SRS>
      <LatLongBoundingBox minx="-179.14734" miny="17.884813" maxx="179.77847" m
    </FeatureType>
  </FeatureTypeList>
```

DescribeFeatureType

- Once you find a layer that looks interesting, you can ask the service to describe it:

```
http://localhost:8888/geoserver/wfs?
version=1.0.0&
service=WFS&
request=DescribeFeatureType&
typename=us_states_poly
```

DescribeFeatureType

- And here is the response:

```
<xsd:schema targetNamespace="http://www.ionicsoft.com/wfs" elementFormDefault="qualified" version="0.1">
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://www.opengis.net/namespaces/gml/core/feature.xsd"/>
  <xsd:element name="us_states_poly" substitutionGroup="gml:_Feature" type="wfs:us_states_poly"> </xsd:element>
- <xsd:complexType name="us_states_poly">
  - <xsd:complexContent>
    - <xsd:extension base="gml:AbstractFeatureType">
      - <xsd:sequence>
          <xsd:element name="AREA" minOccurs="0" nillable="true" type="xsd:double"> </xsd:element>
          <xsd:element name="PERIMETER" minOccurs="0" nillable="true" type="xsd:double"> </xsd:element>
          <xsd:element name="ST99_D00_" minOccurs="0" nillable="true" type="xsd:int"> </xsd:element>
          <xsd:element name="ST99_D00_I" minOccurs="0" nillable="true" type="xsd:int"> </xsd:element>
          <xsd:element name="STATE" minOccurs="0" nillable="true" type="xsd:string"> </xsd:element>
          <xsd:element name="NAME" minOccurs="0" nillable="true" type="xsd:string"> </xsd:element>
          <xsd:element name="LSAD" minOccurs="0" nillable="true" type="xsd:string"> </xsd:element>
          <xsd:element name="REGION" minOccurs="0" nillable="true" type="xsd:string"> </xsd:element>
          <xsd:element name="DIVISION" minOccurs="0" nillable="true" type="xsd:string"> </xsd:element>
          <xsd:element name="LSAD_TRANS" minOccurs="0" nillable="true" type="xsd:string"> </xsd:element>
          <xsd:element name="GEOMETRY" minOccurs="0" nillable="true" type="gml:PolygonPropertyType"> </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

GetFeature

- Finally, you can request the data layer:

```
http://localhost:8888/geoserver/wfs?
version=1.0.0&
service=WFS&
request=GetFeature&
typename=us_states_poly
```

GetFeature

- And here is the response:
  - This is called GML (Geography Markup Language)

```
<gml:featureMember>
 - <au1:us_states_poly fid="us_states_poly.165">
     <au1:AREA>2.8039198120510 06</au1:AREA>
     <au1:PERIMETER>2.201919233137796</au1:PERIMETER>
     <au1:ST99_D00_>167</au1:ST99_D00_>
     <au1:ST99_D00_I>166</au1:ST99_D00_I>
     <au1:STATE>08</au1:STATE>
     <au1:NAME>Colorado</au1:NAME>
     <au1:LSAD>01</au1:LSAD>
     <au1:REGION>4</au1:REGION>
     <au1:DIVISION>8</au1:DIVISION>
     <au1:LSAD_TRANS/>
   - <au1:GEOMETRY>
     - <gml:Polygon srsName="EPSG:4326">
       - <gml:outerBoundaryIs>
         - <gml:LinearRing srsName="EPSG:4326">
           - <gml:coordinates>
               -107.918421,41.002036 -107.69133582431 42,41.00210
               -107.573624,41.00231513722544 -107.5215053632723
               -107.317794462401 12,41.00296721336712 -107.30531
               -107.30252872253156,41.002934686503345 -107.2411
               -107.1160809891758,41.00313681928827 -107.000606
               -106.453859,41.002057 -106.439563,41.001978 -106.43
```

(…Astoundingly boring, isn't it?)

Example KML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Placemark>
  <name>New York City</name>
  <description>New York City</description>
  <Point>
    <coordinates>-74.006393,40.714172,0</coordinates>
  </Point>
</Placemark>
</kml>
```

The MIME type associated with KML is *application/vnd.google-earth.kml+xml*; the MIME type associated with KMZ is *application/vnd.google-earth.kmz*.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark>
      <name>CDATA example</name>
      <description>
        <![CDATA[
          <h1>CDATA Tags are useful!</h1>
          <p><font color="red">Text is <i>more readable</i> and
          <b>easier to write</b> when you can avoid using entity
          references.</font></p>
        ]]>
      </description>
      <Point>
        <coordinates>102.595626,14.996729</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Style id="transBluePoly">
      <LineStyle>
        <width>1.5</width>
      </LineStyle>
      <PolyStyle>
        <color>7dff0000</color>
      </PolyStyle>
    </Style>
    <Placemark>
      <name>Building 41</name>
      <styleUrl>#transBluePoly</styleUrl>
      <Polygon>
        <extrude>1</extrude>
        <altitudeMode>relativeToGround</altitudeMode>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates> -122.0857412771483,37.42227033155257,17
              -122.0858169768481,37.42231408832346,17
              -122.085852582875,37.42230337469744,17
              -122.0858799945639,37.42225686138789,17
              -122.0858860101409,37.4222311076138,17
```


Bowdoin
Peary-MacMillan Arctic Museum
Chapel
Bannister Hall
Morrell Gymnasium
Visual Arts Ce
Massachusetts Hall
isitor Parking
Searle
Memorial Hall – Pickard The
Adams Hall
Image © 2006 TerraMetrics
Google
Pointer 43°54'32.59" N 69°57'43.31" W elev 69 ft
Streaming 100%
Eye alt 479 ft

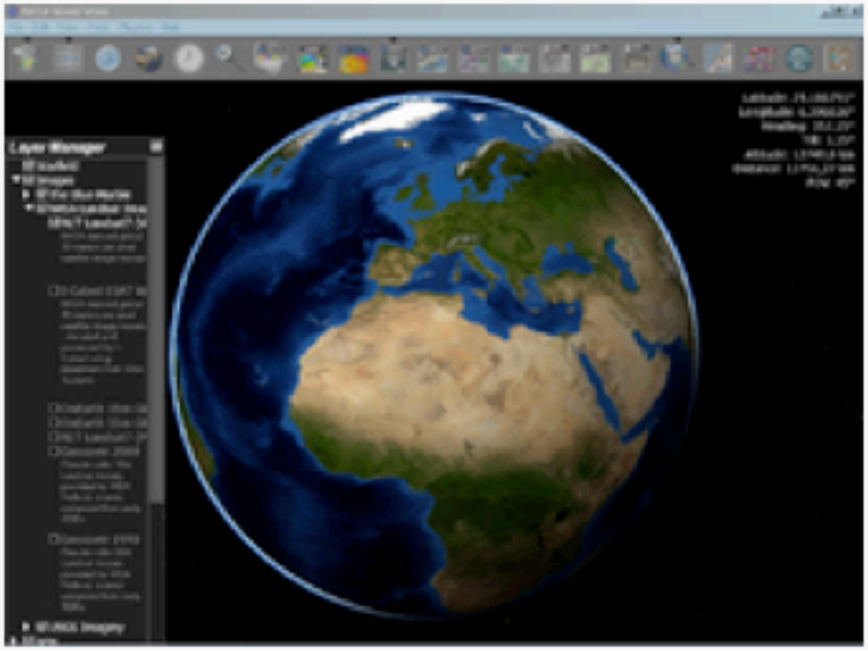
NASA World Wind

From Wikipedia, the free encyclopedia

(Redirected from Nasa worldwind)

World Wind is a free,[1] open source virtual globe developed by NASA and the open source community for use on personal computers. Old versions need Microsoft Windows but more recent Java versions are cross platform. The program overlays NASA and USGS satellite imagery, aerial photography, topographic maps and publicly available GIS data on 3D models of the Earth and other planets.

NASA World Wind



Screenshot of World Wind showing Blue Marble Next Generation layer

Developer(s)	NASA Ames Research Center
Initial release	2004
Stable release	1.4 / February 14, 2007
Written in	C#, Java
Operating system	Cross platform
Available in	English
Type	Virtual globe

[The 2009 JavaOne] Scriptbowl had 5 contenders: Jython, Groovy, Clojure, Scala and JRuby.

...via twitter: "Groovy won the Script Bowl at #javaone thanks to the griffon team for the demo".

Here are a few snapshots of the demo: a mashup of NASA WorldWind and TwitterAPI, packaged as a Griffon application (Twittersphere):


twittersphere
jboner
Doing my JavaOne talk on alternative concurrency paradigms tomorrow at JavaForum in Stockholm 17:00: http://is.gd/12N1w
everflux
Back from javaone and San francisco – big jetlag and too many ideas. Flight was great, thnx @lufthansa
1000 Km
Animated
Search
Results:
20
griffon
Search

NeoGeography

[This term] arose with the concept of Web 2.0, around the increased public appeal of mapping and geospatial technologies that occurred with the release of such tools as "slippy maps" such as Google Maps, Google Earth, and also with the decreased cost of geolocated mobile devices such as GPS units.

http://en.wikipedia.org/wiki/Neogeography

Ajax Maps

vs.

RIA Maps

Rich Internet application

From Wikipedia, the free encyclopedia

Rich Internet applications (**RIA**s) are web applications that have most of the characteristics of desktop applications, typically delivered by way of standards based web browser plug-ins or independently via sandboxes or virtual machines.[1] Examples of RIA frameworks include Curl, GWT, Adobe Flash/Adobe Flex/AIR, Java/JavaFX,[2] uniPaaS[3], Mozilla's XUL and Microsoft Silverlight.[4]

Adobe Flex

From Wikipedia, the free encyclopedia

Adobe Flex is a software development kit released by Adobe Systems for the development and deployment of cross-platform rich Internet applications based on the Adobe Flash platform. Flex applications can be written using Adobe Flex Builder or by using the freely available Flex compiler from Adobe.

The initial release in March 2004 by Macromedia included a software development kit, an IDE, and a J2EE integration application known as Flex Data Services. Since Adobe acquired Macromedia in 2005, subsequent releases of Flex no longer require a license for Flex Data Services, which has become a separate product rebranded as LiveCycle Data Services.

In February 2008, Adobe released the Flex 3 SDK under the open source Mozilla Public License. Adobe Flash Player, the runtime on which Flex applications are viewed, and Adobe Flex Builder, the IDE built on the open source Eclipse platform and used to build Flex applications, remain proprietary.

Obtaining and Installing the Adobe Flex SDK

Incorporating Google Maps into a Flex application requires understanding not only ActionScript code but Flex MXML files. The examples in this documentation use the freely available Adobe Flex 3 SDK available at the following URL address:

http://www.adobe.com/products/flex/overview/#section-3

Note: The Flex SDK requires Java 1.4 or 1.5 (depending on the development platform). Full requirements for development with the Flex SDK are available at:

http://www.adobe.com/products/flex/systemreqs/

Once you've downloaded the SDK, unzip it into a working directory and ensure that you can execute the MXML Compiler (`mxmlc`) from the command line. The `mxmlc` compiler is located in the Flex SDK's `bin` directory. Executing the compiler with the `-help` option should result in the following output.

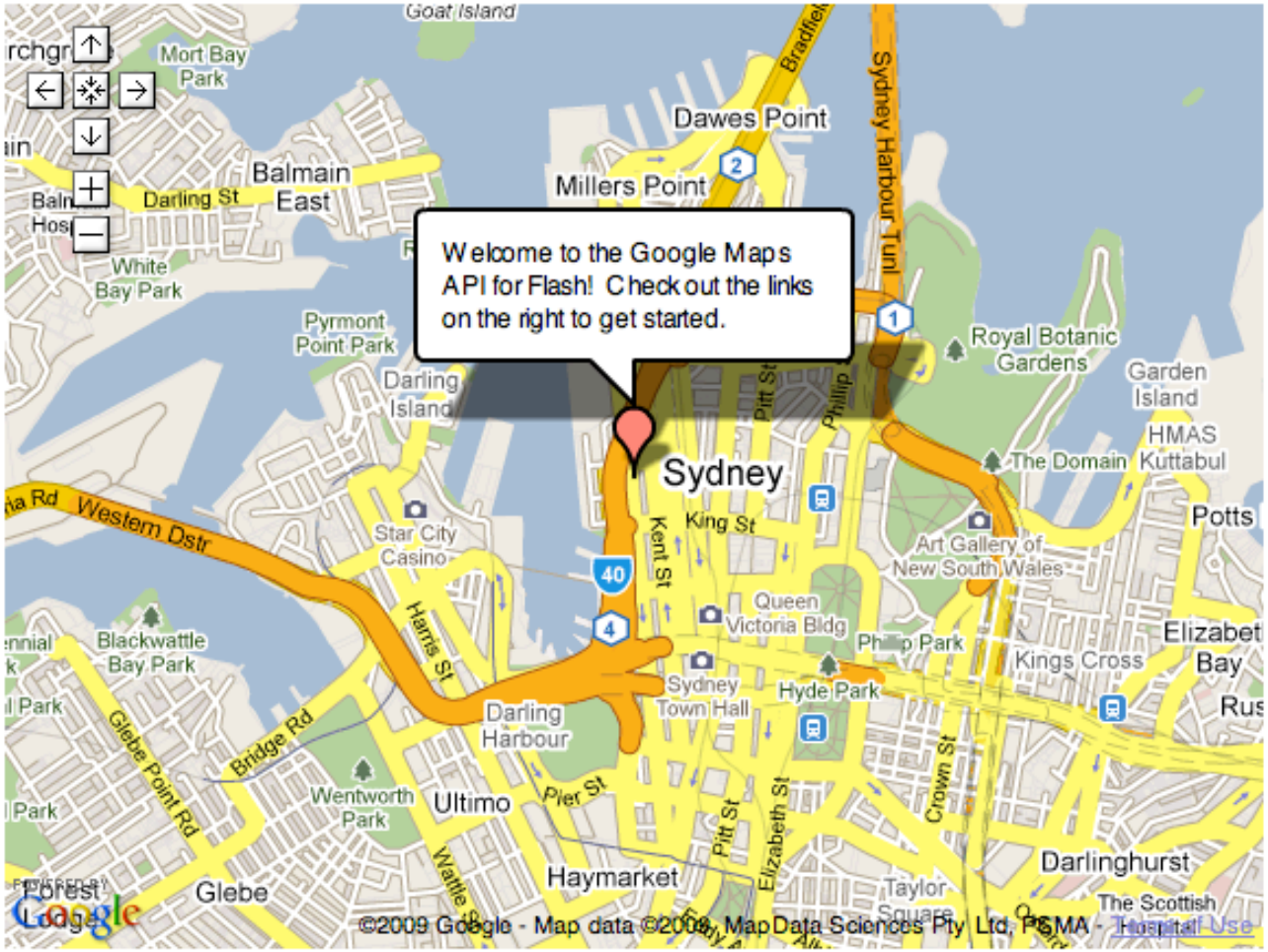
```
hostname$ cd flex_sdk/bin
hostname$ ./mxmlc -help
Adobe Flex Compiler (mxmlc)
Version 3.0.0 build 477
Copyright (c) 2004-2007 Adobe Systems, Inc. All rights reserved.
...
```

You may also want to add the `flex_sdk/bin` directory to your `$PATH` so that you can execute the compiler from any directory.

What is the Google Maps API for Flash?

This API lets Flex developers embed Google Maps in Flash applications. Similar to the JavaScript version, this ActionScript API provides a number of utilities for manipulating and adding content to maps through a variety of services, enabling you to embed robust, interactive maps applications on your website.

New! The Google Maps API for Flash now supports 3D Maps.



Obtaining the Interface Library

Developing Flash content that integrates Google Maps requires inclusion of a Google Maps API for Flash interface library within your application code. This library consists of a `*.swc` file within the `lib` directory of the Maps API for Flash SDK available at the following URL:

http://maps.googleapis.com/maps/flash/release/sdk.zip

The SDK includes two `SWC` files: a Flex version for use within FlexBuilder (or with the free Flex SDK), and a non-Flex version for use within Flash CS3. The Flex `*.swc` is denoted with a `_flex` suffix in the filename.

These SWC files contain interfaces for all public classes in the Google Maps API for Flash development environment. Compiling your application with this library ensures that it can utilize and communicate with all public functionality of the runtime Google Maps API for Flash library, which is retrieved from Google's servers whenever a client loads your application.

MXML Declarations

The MXML declaration defines UI elements within a Flex application, while embedded ActionScript code within the `<mx:Script>` tag defines actions on those UI elements. In the simplest case, you simply declare a `com.google.maps.Map` object within MXML and initialize the map with ActionScript code:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
<maps:Map xmlns:maps="com.google.maps.*" id="map" mapevent_mapready="onMapReady(event)"
  width="100%" height="100%" key="your_api_key"/>
<mx:Script>
    <![CDATA[

    import com.google.maps.LatLng;
    import com.google.maps.Map;
    import com.google.maps.MapEvent;
    import com.google.maps.MapType;

    private function onMapReady(event:Event):void {
      this.map.setCenter(new LatLng(40.736072,-73.992062), 14, MapType.NORMAL_MAP_TYPE);
    }
    ]]>
</mx:Script>
</mx:Application>
```

Compiling Your SWF File

We now have a `HelloWorld.mxml` within our source's root directory and ActionScript code within that file's `<mx:Script>` object. We are ready to compile our code into a SWF (Shockwave Flash) file. We do so using the Flex SDK's `mxmlc` compiler.

The `mxmlc` compiler requires a target `*.mxml` file. As well, to use the Google Maps API for Flash, we use the compiler's `-library-path` parameter to add the location of the Google Maps API for Flash Library (`maps_flex_1_7.swc`) provided to you. A sample invocation appears below:

```
# Execute this command from the ROOT of your development directory, not from within
# the leaf of the namespace (e.g. not in com/google/maps/examples)
#
hostname$:~/src/helloworld$ path_to_flex_sdk/bin/mxmlc HelloWorld.mxml -library-path+=maps_f
lex_1_7.swc
Loading configuration file /home/src/flex_sdk/frameworks/flex-config.xml
/home/src/helloworld/HelloWorld.swf (22223 bytes)
```

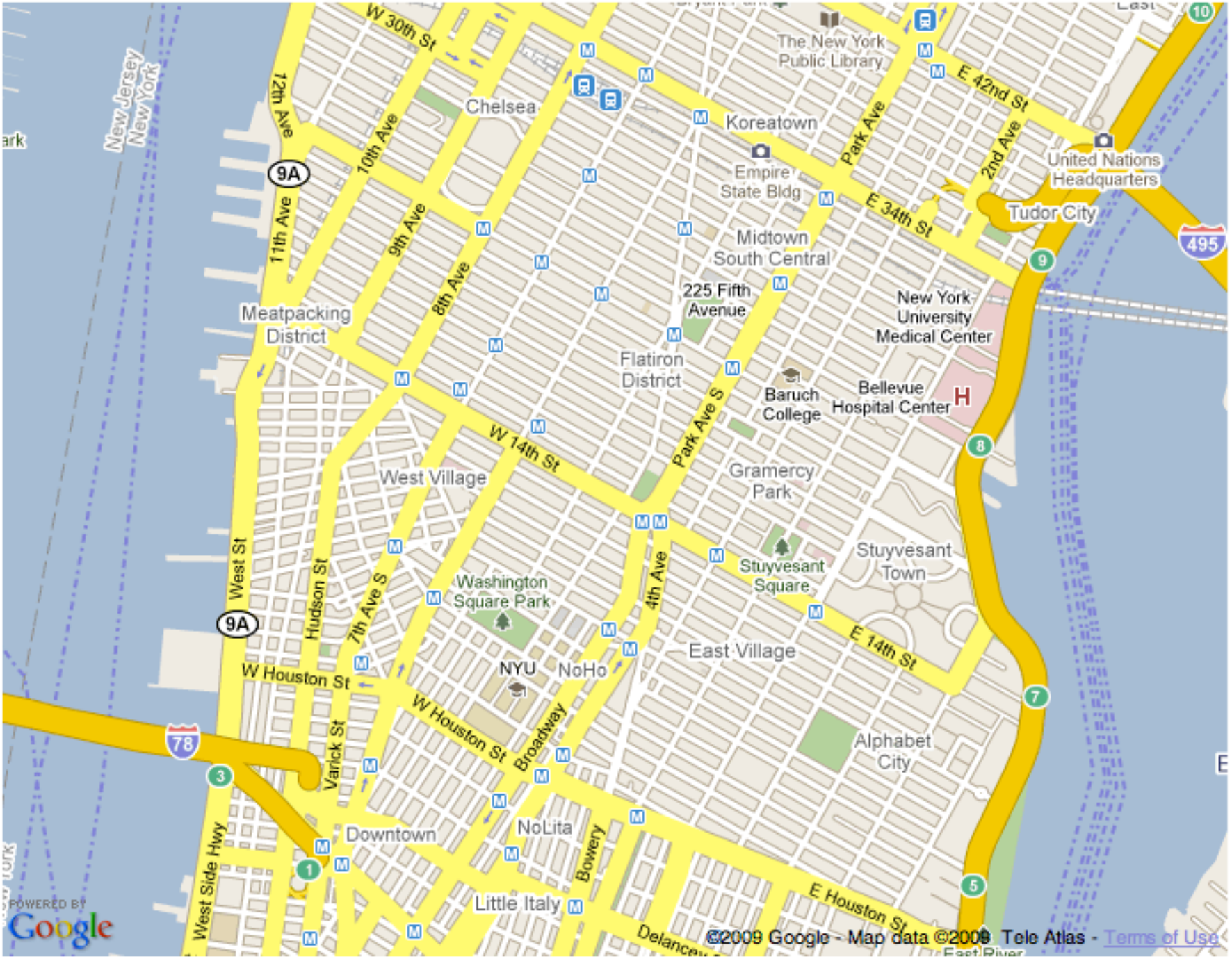
Hosting Your SWF File in a Web Page

Google Maps Flash SWF files, if they are compiled with the API key included, can simply be displayed as standalone files. This is useful for testing, but may not be practical for proper page layout. As a result, you will likely want to set up an HTML page to contain the SWF file. To ensure your SWF file executes within both Internet Explorer and other browsers, you should add the SWF within both `object` and `embed` tags.

A simple HTML page that contains our `HelloWorld.swf` file appears below. For the map to display on a web page, we must reserve a spot for it. We do so in this example by creating a named `div` element and adding the `object` element to it.

```
  <div id="map_canvas" name="map_canvas">
    <object
      classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
      codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=
6,0,29,0"
      width="800px"
      height="600px">
      <param name="movie" value="helloworld.swf">
      <param name="quality" value="high">
      <param name="flashVars" value="key=your_api_key">
      <embed
        width="800px"
        height="600px"
        src="helloworld.swf"
        quality="high"
        flashVars="key=your_api_key"
        pluginspage="http://www.macromedia.com/go/getflashplayer"
        type="application/x-shockwave-flash">
      </embed>
    </object>
  </div>
```

The resulting HTML page is shown below.



But wait --

there's more!

Developing Adobe AIR® Applications (Experimental)

The Google Maps API for Flash now not only supports development of Flash applications within the browser, but there's also an experimental feature that supports Adobe AIR® applications running on the desktop. AIR applications provide additional capabilities not available to applications running within a browser, including file and local network access, application state persistence, and data access.

What is AIR?

AIR is a runtime environment that allows applications to run natively within a variety of operating systems from a common code base. An AIR Installer compiles this code into a format usable by the runtime environment. AIR applications allow you the freedom to interact directly with the desktop, providing file access, data storage, and robust user interface components across operating systems.

More information on Adobe AIR is located at http://www.adobe.com/products/air/.

Code Changes for AIR Applications

Creating Google Maps API for Flash AIR applications is similar to creating SWF files for use within a browser with two exceptions:

1. Instead of am `mx:Application`, the root of the AIR application is a `mx:WindowedApplication`. This should be created by default when you create your initial project.
2. `Map` objects within AIR applications properties must have a `url` property set to an online location where the purpose and scope of the application is described. This `url` must be the same URL that your registered when signing up for an API key.

Because AIR applications have access to the file system, they can perform tasks that browser-based applications cannot easily do. We'll add some code to our test application that responds to user clicks by writing the clicked latitude/longitude value to a text file.

But wait --

there's STILL more!

Introduction

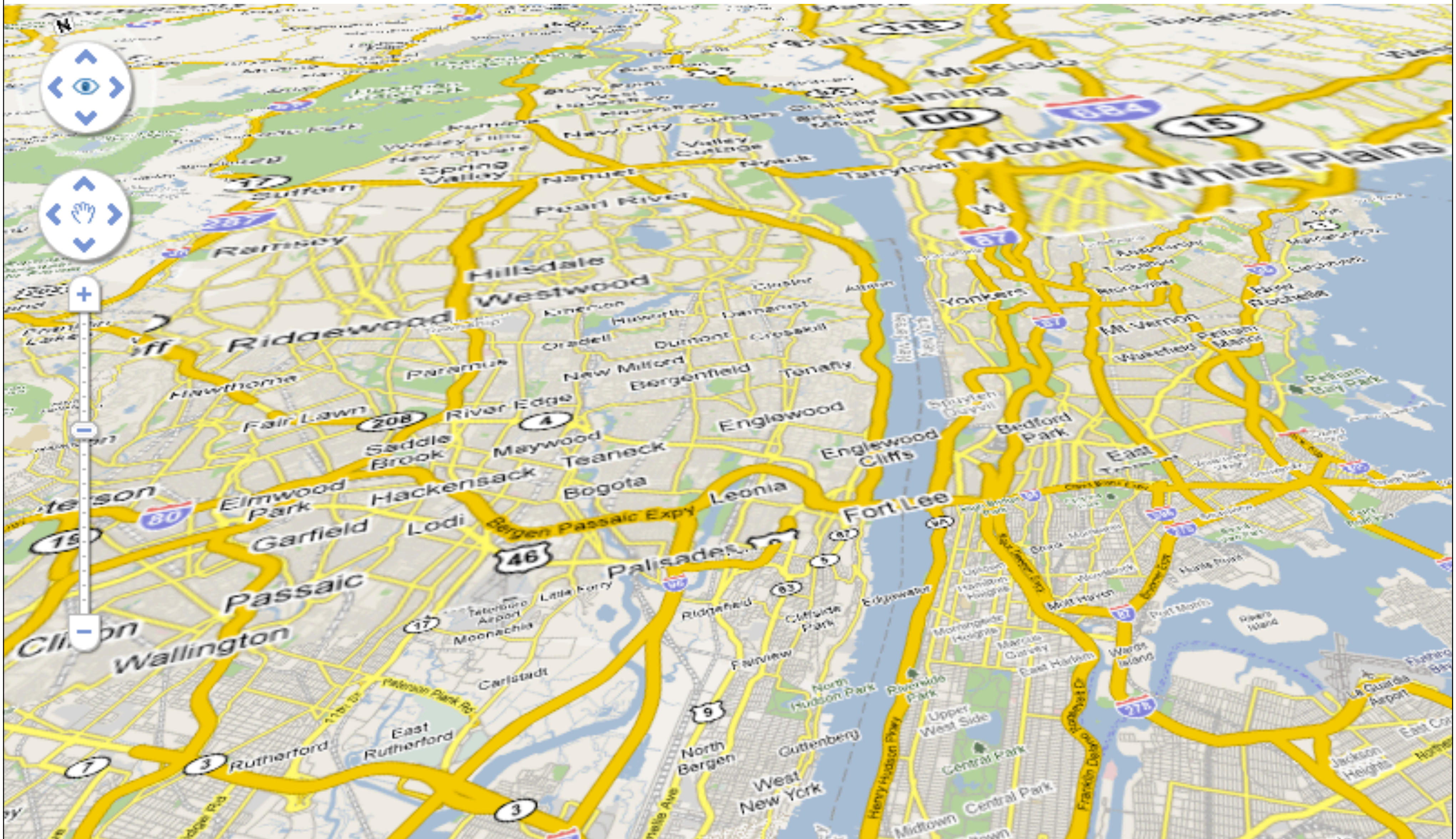
The Google Maps API for Flash now contains support for three-dimensional (3D) maps. 3D maps using the Maps API for Flash are shown using a realistic perspective much like that used within Google Earth.

Use of 3D maps within the Google Maps API for Flash requires use of a new object, the `Map3D` object, which is used in place of the standard `Map` object. Rendering of 3D maps is supported on both Flash 9 and Flash 10 players. However, due to geometry support, use of Flash 10 is recommended as it results in faster rendering. Consult Compiling for Flash 9/10 for more information about how to compile your applications for Flash 9 or 10.

The following code displays a map of New York City, but uses a 3D map and initializes it with an oblique angle:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
 <maps:Map3D xmlns:maps="com.google.maps.*" mapevent_mappreinitialize="onMapPreinitialize(ev
ent)" id="map"
        width="100%" height="100%" key="abc"/>
<mx:Script>
  <![CDATA[
  import com.google.maps.LatLng;
  import com.google.maps.Map3D;
  import com.google.maps.MapEvent;
  import com.google.maps.MapOptions;
  import com.google.maps.MapType;
  import com.google.maps.View;
  import com.google.maps.geom.Attitude;

  private function onMapPreinitialize(event:MapEvent):void {
    var myMapOptions:MapOptions = new MapOptions();
    myMapOptions.zoom = 12;
    myMapOptions.center = new LatLng(40.756054, -73.986951);
    myMapOptions.mapType = MapType.NORMAL_MAP_TYPE;
    myMapOptions.viewMode = View.VIEWMODE_PERSPECTIVE;
    myMapOptions.attitude = new Attitude(20,30,0);
    map.setInitOptions(myMapOptions);
  }
  ]]>
</mx:Script>
</mx:Application>
```


White Plains
Ramsey
Hillsdale
Westwood
Ridgewood
Paramus
Hawthorne
Fair Lawn
River Edge
New Milford
Bergenfield
Tenafly
Dumont
Oradell
Haworth
Demarest
Cresskill
Englewood
Englewood Cliffs
Saddle Brook
Maywood
Teaneck
Hackensack
Elmwood Park
Garfield
Lodi
Bogota
Leonia
Fort Lee
Bergen Passaic Expy
Palisades
Passaic
Wallington
Ridgefield
Cliffside Park
Fairview
Carlstadt
Moonachie
Rutherford
East Rutherford
North Bergen
West New York
Guttenberg
Yonkers
Mt Vernon
Bedford Park
Pearl River
Nanuet
Spring Valley
Suffern
Tarrytown
Central Park
Upper West Side
Midtown

So, what have we learned?

Desktop - Browser

Deployment - Functionality

Flex - Air

A

TALE OF TWO CITIES.

BY

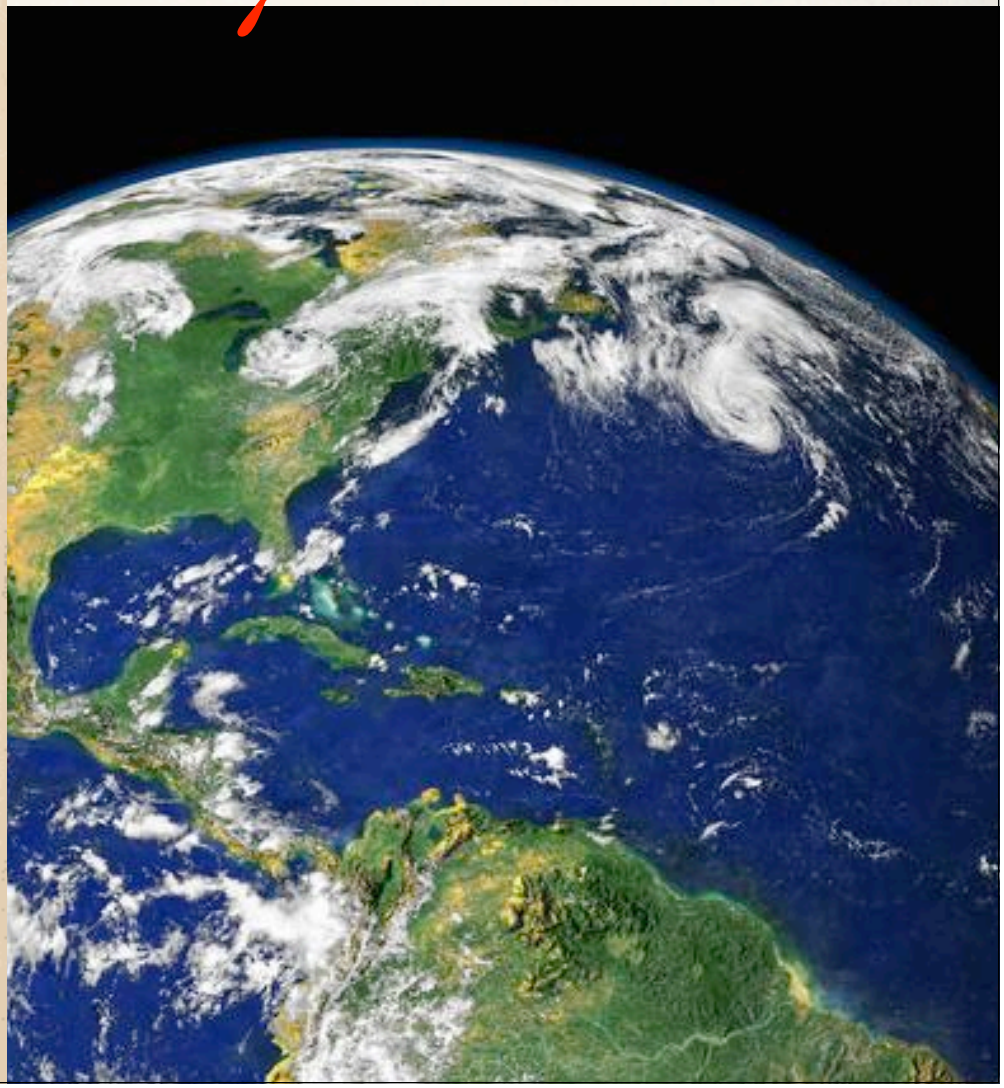
CHARLES DICKENS.

WITH ILLUSTRATIONS BY H. K. BROWNE.

LONDON:
CHAPMAN AND HALL, 193, PICCADILLY;
AND AT THE OFFICE OF ALL THE YEAR ROUND,
11, WELLINGTON STREET NORTH.
MDCCCLIX.

Slippy Maps

(...and Scott Davis)



Scott Davis
scott@thirstyhead.com

Questions?
Thanks for your time.