Geography 485L/585L - Internet Mapping

Karl Benedict - kbene@unm.edu

Spring 2016

Contents

Chapter 1

Week 1 - Module 1 - Introduction and Outline

Overview

- Introductions
- Review of the Syllabus
- Topics to be Covered
- Basics/Definitions

Introductions

- Who am I?
- Who are you?
- What brought you here?

Syllabus Review (link)

Class Topics

- Internet Mapping Clients: Basic HTML, Javascript, CSS; Google Maps API; OpenLayers javascript library
- Geospatial Services Oriented Architectures (SOA)
- Open Standards: Open Geospatial Consortium (OGC - WMS, WFS, WCS, KML); Extensible Markup Language (XML)
- Desktop client use of Open Standards
- Data sharing/publication using Open Standards

Basics

Outline

- What is Internet Mapping?
- Definitions
- Tools

What is Internet Mapping

Extended Desktop Mapping Use of open standards based remote data and map services in desktop applications

Geospatial Data Sharing Establishing open standards based services to share geospatial data and mapping capabilities over the Internet

Web-client Mapping The delivery of mapping and geospatial data tools through web browsers, again based upon open standards

Definitions

Internet The global computer network of computers that typically connect with each other over TCP/IP

World Wide Web The subset of applications that are run over the Internet, typically using the HTTP protocol in combination with data (HTML, XML, XHTML), presentation (CSS), and behavior (JavaScript) components

Mapping The generation of cartographic products that include map images (pictures of geospatial data) and other elements (e.g. legends, tools, scale information, north-arrow)

Definitions

Analysis The development of models (statistical and otherwise) that enable the exploration of geospatial data and testing of hypotheses using those data

Open Standards While the definition varies from one organization to the next, Open Standards are often characterized by the following:

- Developed through a public process by a national or international standards group
- May be implemented royalty-free

Definitions

Interoperability Ability of systems to share data and information with each other

COTS Commercial Off-the-Shelf Software. Applications that are "purchased" from vendors, often with license terms that restrict the use the software to the specific platform for which it is licensed. Often comes with implicit or explicit technical support

Open Source Software licensed under terms that are consistent with the Open Source definition, which includes access to source code, and freedom to modify and redistribute

Definitions

Data Actual values associated with geographic locations. For example - numeric elevation values associated with locations within a Digital Elevation Model.

Metadata Data about a particular data product or service. Metadata provide critical documentation that supports the discovery and use of data products and data and mapping services

Tools

Computer Hardware Requirements

- At least 2 GB RAM
- At least 20 GB of available disk space
- Internet Connection (broadband [>728 Kb/sec] recommended)

Software Requirements

- Supported Operating System
- Geographic Information System (GIS)
- Text Editor
- Secure File Transfer Protocol Client
- Secure Shell (SSH) Client
- Web Browser (at least one of the following)
- A desktop Git/GitHub client for your operating system of choice

Chapter 2

Week 2 - Module 2a - Web-based Mapping Clients. HTML, CSS & Javascript

Overview

- Web Development
- Parts of a web page
- Web Site Components
  - Structure (X/HTML)
  - Presentation (CSS)
  - Behavior (Javascript)
- Simple Web Pages
- More Complete Web Page Example

Web Development

- Requirements
  - Web Server
  - File location that the web server accesses for requested content
  - Files must be readable by all users
- General Process
  - Create basic content in HTML or XHTML (structure)
  - Change appearance of content through the definitions of styles using CSS (presentation)
  - Add dynamic capabilities to content through Javascript (behavior)
  - REPEAT over and over and over and over again

Parts of a Web Page

```
1 <html>
2     <!-- The HTML block is the container for all of your page content -->
3     <head>
```

```
4           <!-- The head is where you include pointers to external resources
5           (i.e. style sheets and javascript files), blocks of Javascript code
6           , styles, etc.  -->
7           <title>The page title also goes in here</title>
8       </head>
9       <body>
10          <!-- The body is where you put all of the content for the page
11          (i.e. the material that will be displayed in the web browser)  -->
12          <h1>Headers</h1>
13          <div>Generic blocks of content</div>
14          <p>Paragraphs</p>
15          <table>Tables</table>
16          <img ...>Images</img>
17          <form ...>Forms</form>
18          <ul>Unordered Lists</ul>
19          <ol>Ordered Lists</ol>
20          <li>List Items</li>
21
22          <!-- Javascript can go here as well -->
23      </body>
24  </html>
```

Link to example/Preview

Web Site Components - Structure

Content is defined in terms of the structural elements available in HTML/XHTML

- Sample HTML/XHTML Tags
  - Paragraphs (i.e. blocks of text) are contained within `<p>...</p>` tags
  - Headings (i.e. section headings, sub-headings) are contained within numerically defined header tags: `<h1>...</h1>, <h2>...</h2>, <h3>...</h3>`, etc.
  - Tabular data are within `<table>...</table>` tags
  - List are specified within `<ol>...</ol>` or `<ul>...</ul>` tags, depending upon whether the list is ordered (numbered) or unordered (e.g. bulleted)
  - User input elements are put within `<form>...</form>` tags
  - Blocks of content (i.e. sections or divisions) are defined within `<div>...</div>` tags
- Structure is translated into the Document Object Model (DOM) for later use by CSS and Javascript

Web Site Components - Presentation

Modifications to default rendering of HTML/XHTML elements are made through styles defined in CSS

- Styles may be
  - defined in an external file that is referenced within the `<head>` block (the preferred method when doing "real" web development)
  - directly defined within the `<head>` block of a web page
  - directly embedded in the elements to which they apply (generally not a "Good Thing")
- When not embedded within an element, a style definition consists of

  - A selector
  - The style definition, enclosed in “curly-brackets”, separated by “semi-colons”
  - For example: `h1 {color:red; font-size:18px;}`

CSS Selectors

Selectors may be based on several criteria

- Element name: `h1`, `p`, `table`, `ul`, etc.
  - Element: `<h1>A top level heading</h1>`
  - Selector: `h1 {color:red; font-size:18px}`
- Element ID: a unique name assigned to HTML/XHTML elements within the structure of the document
  - Element: `<p id="para01">Some text goes here</p>`
  - Selector: `#para01 {color:blue; font-size:12px}`
- Class ID: a name assigned to multiple elements which may be modified through reference to their class
  - Element: `<p class="instructions">Here are some instructions</p>`
  - Another Element: `<p class="instructions">Here are some more instructions</p>`
  - Selector: `.instructions {color:red; font-size:12px; text-decoration:blink}`
- Selectors may be combined in a variety of ways

Web Site Components - Behavior

The most interoperable language for adding dynamic behavior to web sites is *Javascript* - supported by most browsers on most operating systems

- A full-fledged programming language
  - A non-trivial undertaking to become proficient in
  - Experience in other programming languages can contribute to learning Javascript
- Defines actions that may be taken on/by DOM elements
- Allows for modification of existing DOM elements, creation of new DOM elements after the page has finished loading from the server, retrieval of new content after page loads
  - An interactive web page that may behave like a local desktop application

Reference Links

- w3schools.com
  - HTML 4.0 / XHTML 1.0 Tag Reference
  - Cascading Style Sheet (CSS) selectors and elements
  - Javascript reference
- World Wide Web Consortium (W3C)
  - HTML and CSS Background
  - HTML and CSS Tutorial Links Page
  - Validators Page
- Webmonkey.com
  - HTML Cheat Sheet
  - CSS Guide

Simple Web Page

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title>This is a simple web page</title>
    </head>
    <body>
        <h1>They don't get any simpler than this!</h1>
        <p>OK, not much simpler than this.</p>
        <p>Hello World?</p>
    </body>
</html>
```

link to example/Preview

Simple Web Page with CSS

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title>This is a simple web page - with styling</title>
        <style type="text/css">
            h1 {color:blue; font-size:large}
            p.para {color:#777777; font-size:small}
            #annoying {color:red; text-decoration:line-through}
        </style>
    </head>
    <body>
        <h1>They don't get any simpler than this!</h1>
        <p class="para">OK, not much simpler than this.</p>
        <p id="annoying" class="para">Hello World?</p>
    </body>
</html>
```

link to example/Preview

Simple Web Page with Javascript

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title>This is a simple web page with Javascript</title>
        <script type="text/javascript">
            function genericAlert() {
                alert("You just did something ...")
                document.getElementById("clickMe").style.color = "red"
            }
        </script>
    </head>
    <body>
        <h1>They don't get any simpler than this!</h1>
```

```
        <p>OK, not much simpler than this.</p>
        <p>Hello World?</p>
        <p id="clickMe" onclick="genericAlert();">What happens when you click me?</p>
    </body>
</html>
```

link to example/Preview

More Complete Web Page Example

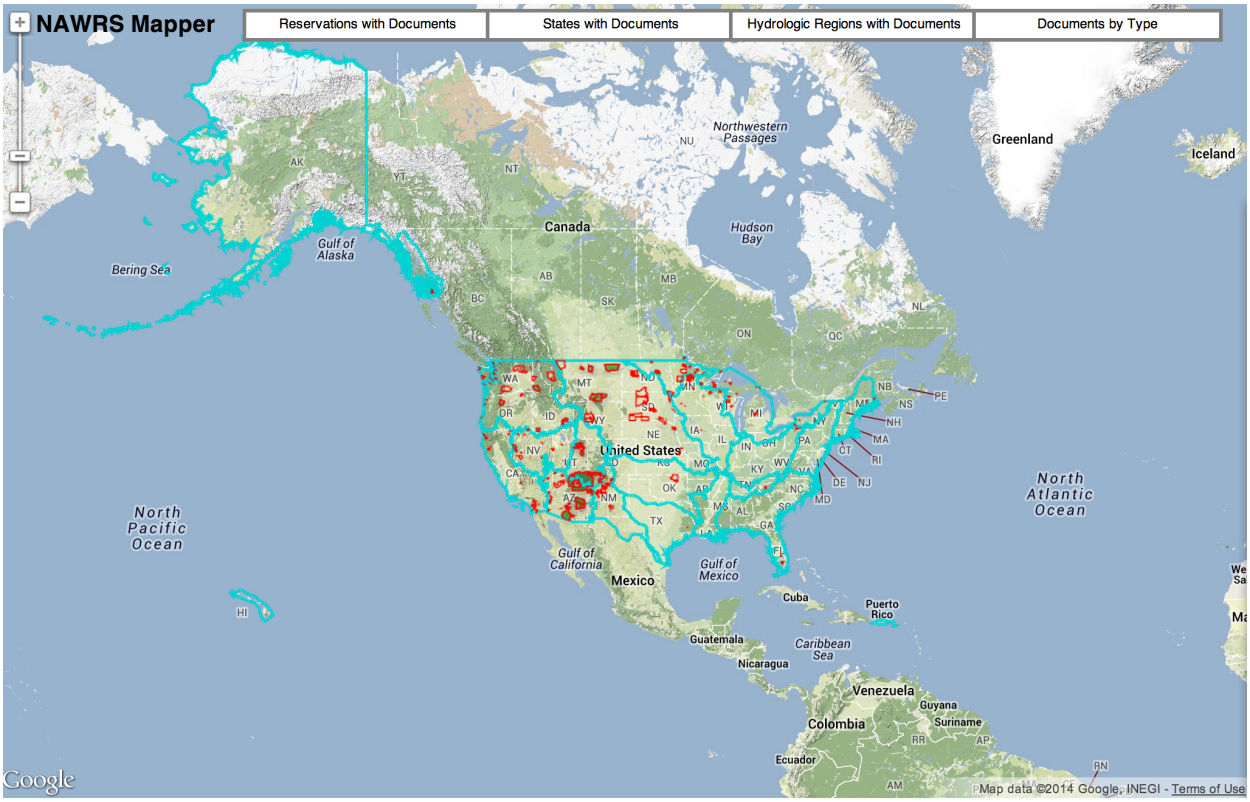


Figure 2.1: NAWRS Mapper. *HTML*: 39 Lines; *CSS*: 136 Lines; *core.js*: 515 Lines + Google Maps API and JQuery Framework

Chapter 3

Week 3- Module 2a - Web-based Mapping Clients. Google Maps API

Outline

- What is an API
- The Google Maps API
  - Version
  - Reference Information
  - Key Components
  - Examples

What is an API

- API Stands for Application Programming Interface

  An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers. – From Wikipedia: http://en.wikipedia.org/wiki/Api

- The Google Maps API provides an interface for interacting with Google's mapping services from external web applications

Google Maps API Version

- The version of the Google Maps API used in this class is v3 of the Javascript API
  - Freely usable for free applications
  - Subject to Google's Terms of Service
  - Google API key is optional for our work
- Key capabilities in v3
  - Interactive maps based on Google's mapping engine (contrast w. static maps API)
  - Optimized for desktop and mobile platforms and applications

Reference Information

Google Maps API Family http://code.google.com/apis/maps/
Javascript API Home Page http://code.google.com/apis/maps/documentation/javascript/
Javascript API v3 Tutorial Page http://code.google.com/apis/maps/documentation/javascript/tutorial.html

Key Components

- Map object options

Types (required) ROADMAP
SATELLITE
HYBRID
TERRAIN
Latitude and Longitude (required) specification of where the map should initially be centered
Zoom Level (required) 0=global, higher values increasingly local. Limited by map type

Controls

- Available Controls (enabled through map options) default controls
  - Zoom Control
  - Pan Control
  - Scale Control
  - MapType Control
  - Street View Control
  - Rotate (for maps that contain 45-degree imagery)
- Different control styles may be defined
- Controls may be positioned positioning options
- Custom controls may be defined and attached to fixed location in the map

Overlays

Overlay Types documentation

Marker points depicted by specified or defined icons at locations within the map
Polyline linear features defined by multiple points with a defined style for the line
Polygon closed features defined by multiple points. Supports multi-polygons, and donuts. Line and fill styles may be specified.
(Ground) Overlay Maps Image-based map layers that replace or overlay Google layers - registered to the map coordinates

Overlays (continued)

Info Windows floating content windows for displaying content defined as HTML, a DOM element, or text string
Layers Grouped display content assigned to a specific layer type: Data (including GeoJSON), KmlLayer (& GeoRSS), Heatmap, FusionTablesLayer, TrafficLayer, TransitLayer BicyclingLayer
Custom Overlays definition of programmatically controlled layers

Services

- Geocoding Service
  - Forward and reverse geocoding:
    - address to LatLon
    - LatLon to Nearest Address
  - May be biased to current viewport, region
- Directions
  - Based upon an origin, destination, and a variety of additional options
  - Available directions and rendered route
- Distance Matrix
  - Travel distance and duration given a specific mode of travel

Services

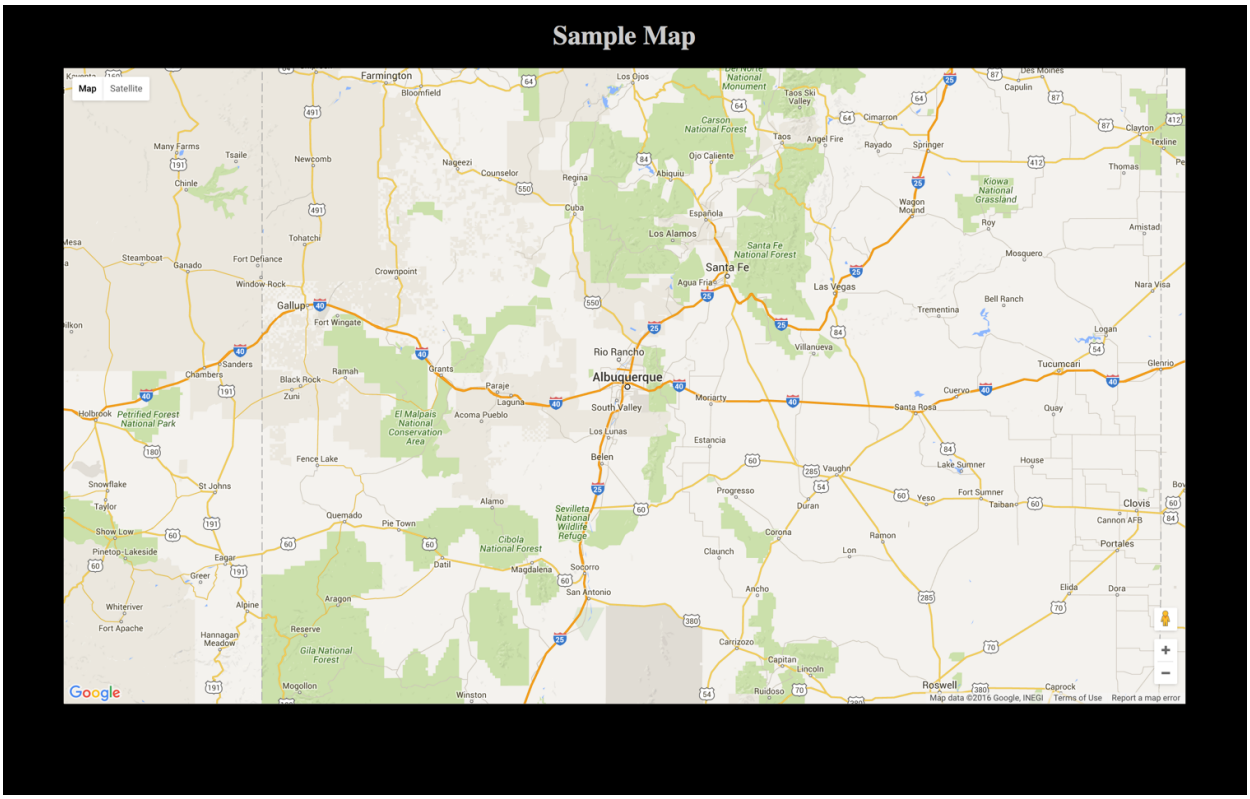
- Elevation
  - Delivery of elevation data for locations or paths
- Streetview
  - Integration of Google Streetview within a DOM element
- Maximum Zoom
  - Provides information about the maximum available zoom level

Events

- Events provide the ability to attach custom behaviors to events in the interface. For example:
  - Changing items in the interface as the user zooms in on a map
  - Displaying additional information outside the map when the user clicks a location in the map
  - Synchronizing the behavior of multiple maps as the user interacts with one map
- Requires higher-level Javascript than we will cover in this course

Examples

Simple - Roadmap



Simple - Roadmap Code

gmaps01.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_01.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
```

```
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

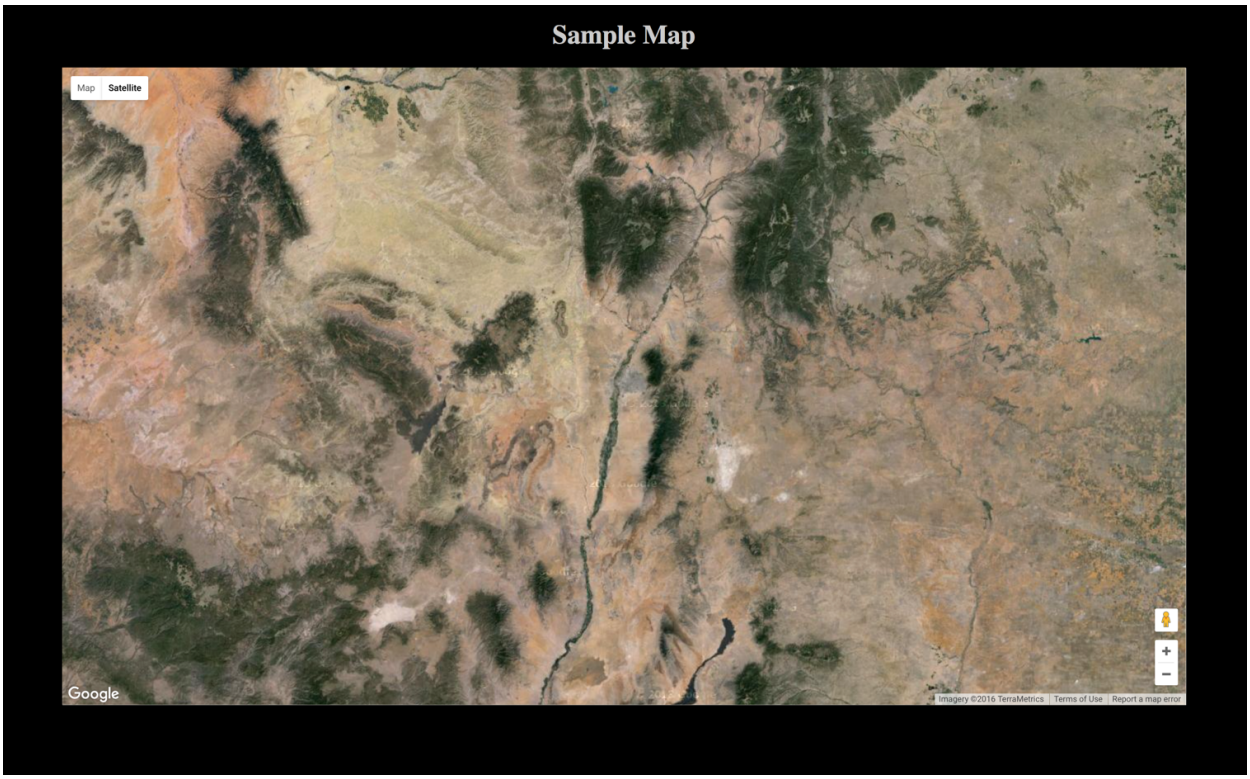
mapPage.css

```
/* You must set the height of either the 'html' or 'body' elements for some
    browsers to properly render the map with a hight taller than 0px */
html {
    height: 100% }

body {
    height: 100%;
    margin: 0px;
    padding: 0px;
    background-color: black;
    color: #CCCCCC;
    text-align: center}

#map_canvas {
    width:90%;
    height:80%;
    margin-left:auto;
    margin-right: auto }

.infoBox {
    color:black }
```

mapPage_01.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var mapOptions = {
        zoom: 8,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        mapOptions);
}

```

Simple - Satellite



Simple - Satellite Code

gmaps02.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_02.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
```

```
    </body>
</html>
```

mapPage_02.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var mapOptions = {
        zoom: 8,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.SATELLITE
    };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        mapOptions);
}

```

Simple - Hybrid

Simple - Hybrid Code

gmaps03.html

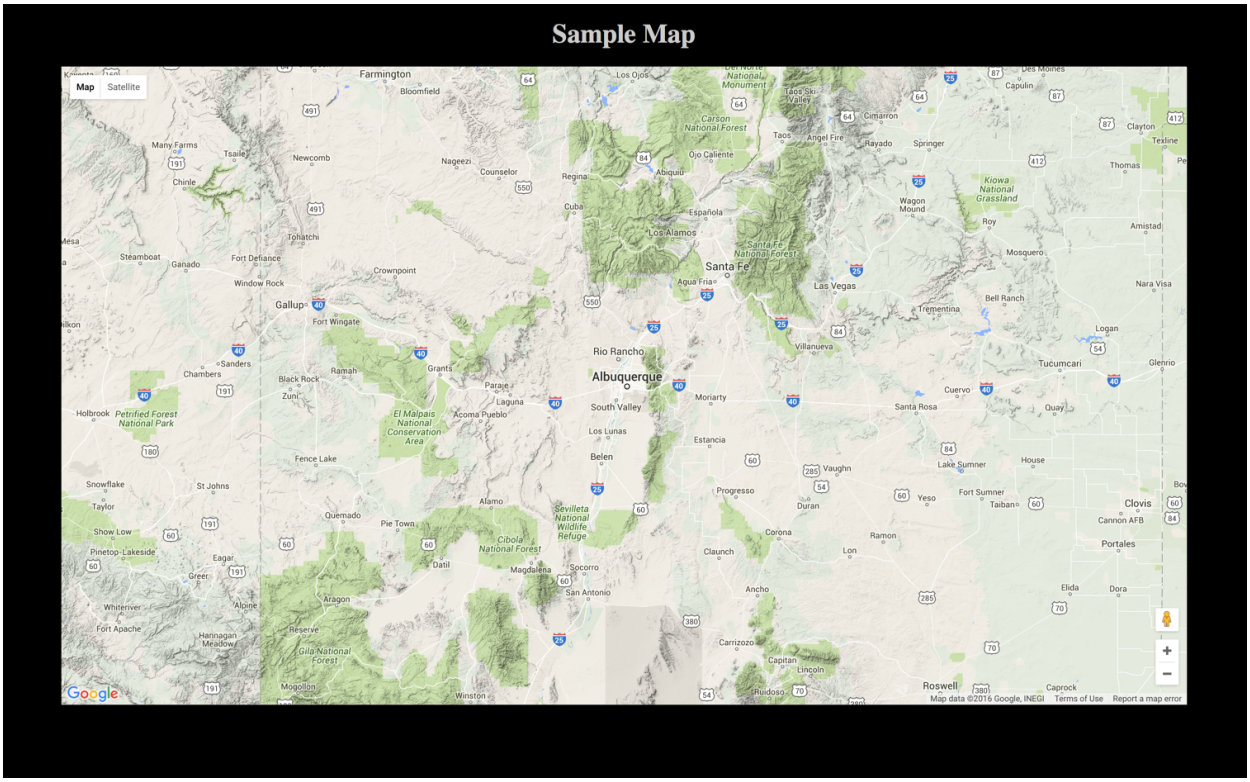
```
<!DOCTYPE html>
<html>
```

```
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_03.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

mapPage_03.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var mapOptions = {
        zoom: 8,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.HYBRID
    };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        mapOptions);
}

```

Simple - Terrain



Simple - Terrain Code

gmaps04.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_04.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
```

```
    </body>
</html>
```

mapPage_04.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var mapOptions = {
        zoom: 8,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.TERRAIN
    };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        mapOptions);
}

```

Simple - Hybrid - Zoomed

Simple - Hybrid - Zoomed Code

gmaps05.html

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_05.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

mapPage_05.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var mapOptions = {
        zoom: 18,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.TERRAIN
    };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        mapOptions);
}

```

Simple - Zoomed - Modified Controls

Simple - Zoomed - Modified Controls Code

gmaps06.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_06.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
```

```
    </body>
</html>
```

mapPage_06.js

```
function initialize() {
var classroom = new google.maps.LatLng(35.084280,-106.624073)
var myOptions = {
    zoom: 18,
    center: classroom,
    mapTypeId: google.maps.MapTypeId.HYBRID,
    zoomControl: true,
    zoomControlOptions: {style: google.maps.ZoomControlStyle.SMALL},
    mapTypeControl: true,
    mapTypeControlOptions: {
    style: google.maps.MapTypeControlStyle.DROPDOWN_MENU},
    streetViewControl: false
};
var map = new google.maps.Map(
    document.getElementById("map_canvas"),
    myOptions);
}

```

Markers

Markers Code

gmaps07.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_07.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

mapPage_07.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var office = new google.maps.LatLng(35.084506,-106.624899)
    var myOptions = {
        zoom: 18,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.HYBRID
        };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        myOptions);

    var classroomMarker = new google.maps.Marker({
        position: classroom,
        title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
        });
    classroomMarker.setMap(map);

    var officeMarker = new google.maps.Marker({
        position: office,
        title:"Office, Bandelier West, Room 107"
        });
    officeMarker.setMap(map);
```

```
}

```

Polyline

Polyline Code

gmaps08.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_08.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
```

```
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

mapPage_08.js

```
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var office = new google.maps.LatLng(35.084506,-106.624899)
    var myOptions = {
        zoom: 18,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.HYBRID
        };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        myOptions);

    var classroomMarker = new google.maps.Marker({
        position: classroom,
        title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
        });
    classroomMarker.setMap(map);

    var officeMarker = new google.maps.Marker({
        position: office,
        title:"Office, Bandelier West, Room 107"
        });
    officeMarker.setMap(map);

    var officeVisitCoordinates = [
        office,
        new google.maps.LatLng(35.084445,-106.624327),
        new google.maps.LatLng(35.084309,-106.624308),
        classroom
        ];
    var officePath = new google.maps.Polyline({
        path: officeVisitCoordinates,
        strokeColor: "#FF0000",
        strokeOpacity: 1.0,
        strokeWeight: 2
    });
    officePath.setMap(map)
}

```

Polygon

Polygon Code

gmaps09.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_09.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
```

```
22      </body>
23  </html>
```

mapPage_09.js

```
1   function initialize() {
2       var classroom = new google.maps.LatLng(35.084280,-106.624073)
3       var office = new google.maps.LatLng(35.084506,-106.624899)
4       var myOptions = {
5           zoom: 18,
6           center: classroom,
7           mapTypeId: google.maps.MapTypeId.HYBRID
8           };
9       var map = new google.maps.Map(
10          document.getElementById("map_canvas"),
11          myOptions);
12      var classroomMarker = new google.maps.Marker({
13          position: classroom,
14          title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
15          });
16      classroomMarker.setMap(map);
17      var officeMarker = new google.maps.Marker({
18          position: office,
19          title:"Office, Bandelier West, Room 107"
20          });
21      officeMarker.setMap(map);
22      var buildingCoordinates = [
23          new google.maps.LatLng(35.084498,-106.624921),
24          new google.maps.LatLng(35.084558,-106.624911),
25          new google.maps.LatLng(35.084566,-106.624970),
26          new google.maps.LatLng(35.084609,-106.624966),
27          new google.maps.LatLng(35.084544,-106.624383),
28          new google.maps.LatLng(35.084438,-106.624317),
29          new google.maps.LatLng(35.084384,-106.623922),
30          new google.maps.LatLng(35.084164,-106.623970),
31          new google.maps.LatLng(35.084214,-106.624324),
32          new google.maps.LatLng(35.084214,-106.624324),
33          new google.maps.LatLng(35.084391,-106.624284)
34          ];
35      var bldgPoly = new google.maps.Polygon({
36          paths: buildingCoordinates,
37          strokeColor: "#FF0000",
38          strokeOpacity: 0.8,
39          strokeWeight: 2,
40          fillColor: "#FF0000",
41          fillOpacity: 0.35
42      });
43      bldgPoly.setMap(map)
44  }
45
```

Adding an Info Window

Adding an Info Window Code

gmaps10.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_10.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize
            function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be
            for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
```

```
    </body>
</html>
```

mapPage_10.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var office = new google.maps.LatLng(35.084506,-106.624899)
    var myOptions = {
        zoom: 18,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.HYBRID
        };
    var map = new google.maps.Map(
        document.getElementById("map_canvas"),
        myOptions);
    var classroomMarker = new google.maps.Marker({
        position: classroom,
        title:"Geography 485L/585L Classroom, Bandelier East, Room 106"
        });
    classroomMarker.setMap(map);
    var officeMarker = new google.maps.Marker({
        position: office,
        title:"Office, Bandelier West, Room 107"
        });
    officeMarker.setMap(map);
    var buildingCoordinates = [
        new google.maps.LatLng(35.084498,-106.624921),
        new google.maps.LatLng(35.084558,-106.624911),
        new google.maps.LatLng(35.084566,-106.624970),
        new google.maps.LatLng(35.084609,-106.624966),
        new google.maps.LatLng(35.084544,-106.624383),
        new google.maps.LatLng(35.084438,-106.624317),
        new google.maps.LatLng(35.084384,-106.623922),
        new google.maps.LatLng(35.084164,-106.623970),
        new google.maps.LatLng(35.084214,-106.624324),
        new google.maps.LatLng(35.084214,-106.624324),
        new google.maps.LatLng(35.084391,-106.624284)
        ];
    var bldgPoly = new google.maps.Polygon({
        paths: buildingCoordinates,
        strokeColor: "#FF0000",
        strokeOpacity: 0.8,
        strokeWeight: 2,
        fillColor: "#FF0000",
        fillOpacity: 0.35
        });
    bldgPoly.setMap(map);
    var classInfoContent = '<div class="infoBox">' +
        '<p>This is the location for the Geography 485L/585L class</p>' +
        '</div>'
    var classInfoWindow = new google.maps.InfoWindow({
        content: classInfoContent
        });
```

```
50      google.maps.event.addListener(classroomMarker, 'click', function() {
51          classInfoWindow.open(map,classroomMarker);
52          });
53  }
54
```

Chapter 4

Week 4 - Module 2a - Web-based Mapping Clients. Google Maps API (pt. 2)

Overview

- Additional Google Maps API Capabilities to be Aware of
  - Styling of the base maps with custom preferences
  - Fusion Tables
- Bringing it all together in a "real" web page

Getting Started with Styled Maps - Video

Styled Maps Documentation | Styled Maps Wizard

Map Example: Simple - Styled

gmap_styled.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/mapPage.css">
    </head>

    <body>
        <h1>Sample Map - Styled (POIs Highlighted)</h1>
        <div id="map_canvas"></div>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_styled.js"></script>

```

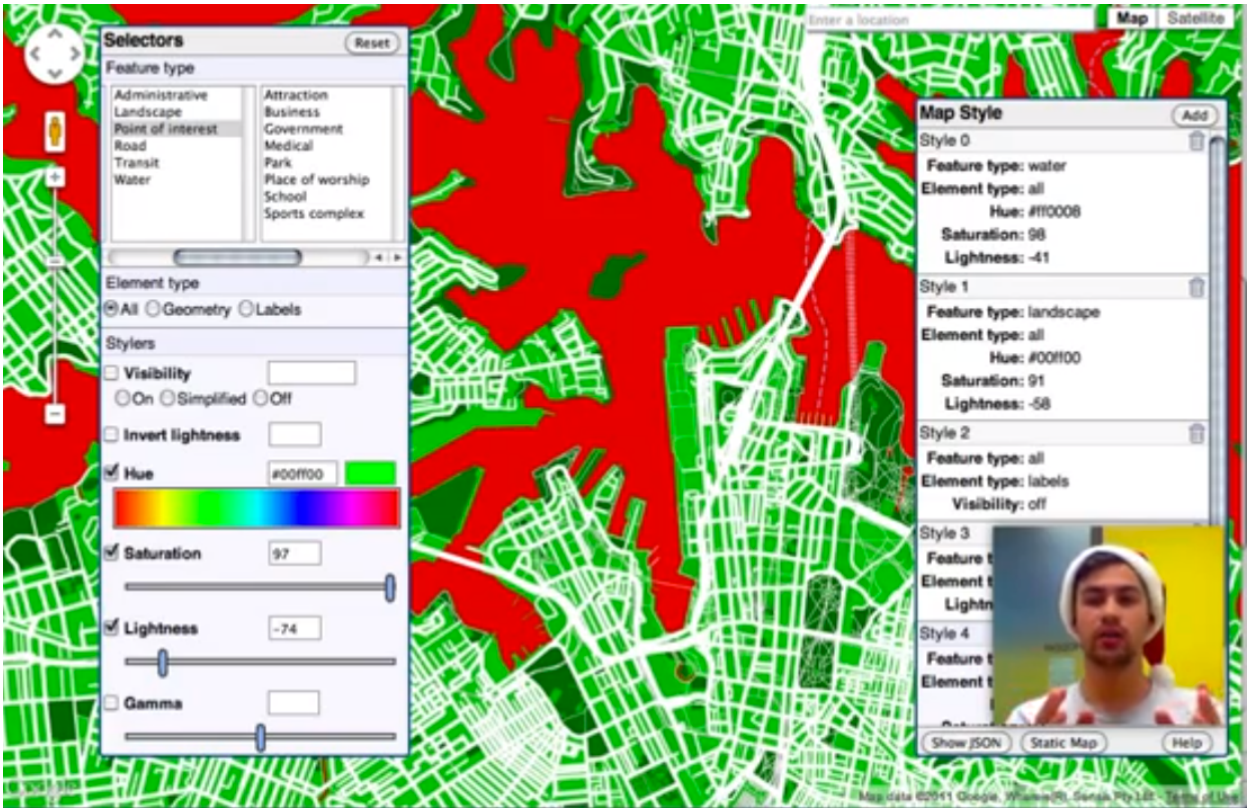


Figure 4.1: Google Maps Styled Maps Wizard link

```
        <!-- Load the API in asynchronous mode and execute the initialize function when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should be for a production map
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

mapPage_styled.js

```
function initialize() {
    var classroom = new google.maps.LatLng(35.084280,-106.624073)
    var myOptions = {
        zoom: 8,
        center: classroom,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        styles: [
          {
            featureType: "water",
            stylers: [
              { visibility: "on" },
              { hue: "#0008ff" }
            ]
          },{
            featureType: "road.highway",
            stylers: [
              { hue: "#ff1a00" }
            ]
          },{
            featureType: "road.arterial",
            stylers: [
              { hue: "#ffa200" },
              { visibility: "simplified" }
            ]
          },{
            featureType: "road.local",
            stylers: [
              { visibility: "off" }
            ]
          },{
            featureType: "administrative",
            stylers: [
              { visibility: "simplified" }
            ]
          },{
            featureType: "poi",
            stylers: [
              { visibility: "on" },
              { hue: "#00ffff" }
            ]
          },{
            featureType: "poi",
            stylers: [
              { visibility: "on" }
```

```
            ]
          }
        ]

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

Google I/O 2011: Managing and visualizing your geospatial data with Fusion Tables - Video

Some particularly relevant sections: Introduction (0:00 - 10:30) | Google Maps API Integration (21:40 - 34:42) | Summary and Links (52:00 52:40)

Fusion Tables Documentation/Help

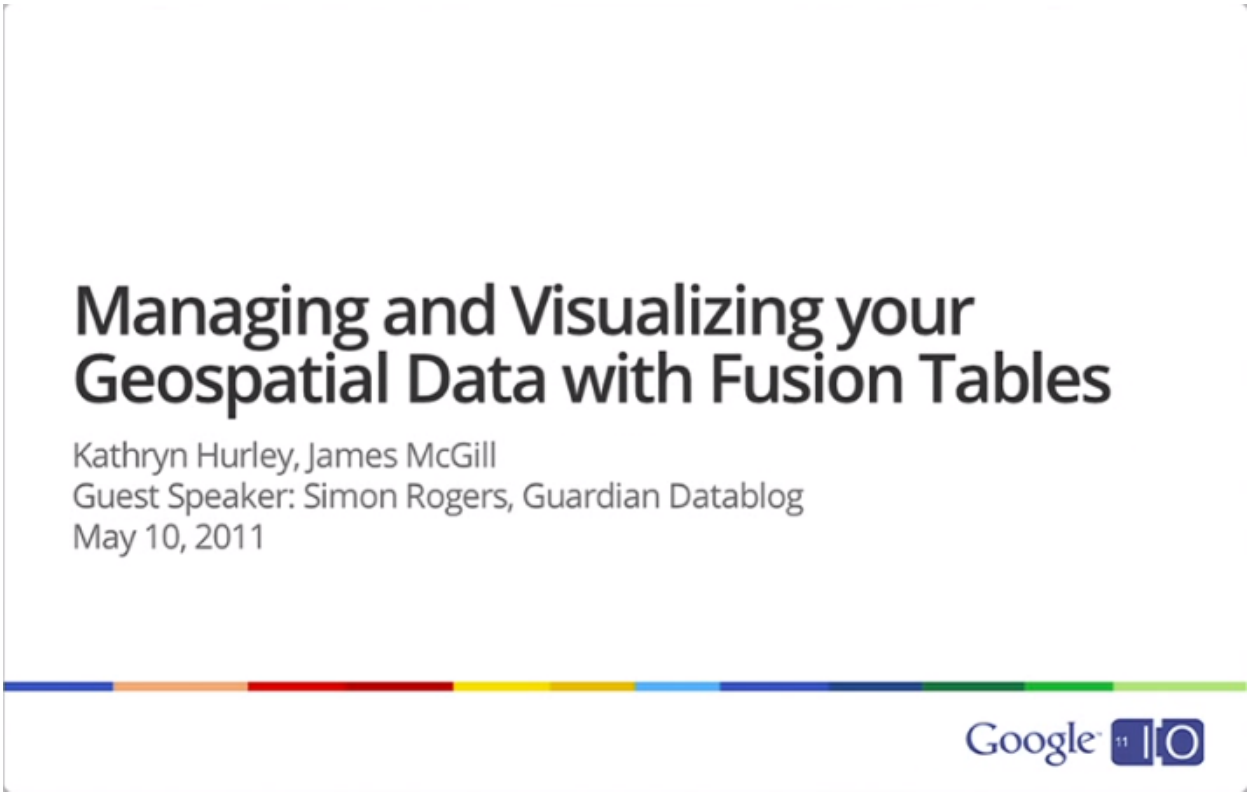


Figure 4.2: Google Fusion Tables Introduction Video link

Bringing It All Together

gmaps_events.html

```
<!DOCTYPE html>
<html>
```

```
<head>
    <link rel="stylesheet" type="text/css" href="css/event_mapPage.css">
    <title>Karl's Event Diary</title>
</head>

<body>
    <h1>My diary of endurance events that I've participated in since joining Team in Training
    </h1>

    <p>In 2008 Cynthia and I joined the Leukemia and Lymphoma Society's
    (<a href="http://www.lls.org/">LLS</a>) Team in Training
    (<a href="http://www.teamintraining.org/">TNT</a>,
    <a href="http://youtu.be/GMSKG8L6K78">info video</a>) program as
    participants to train for the Animas Valley/Steamworks Half Marathon
    and raise money for blood cancer research and patient services. In spite of
    our not having any direct connection to blood cancer (at that time),
    we found the goals of LLS admirable,  the combined training and fund-raising
    program of TNT a great idea, and have made many new friends over the many
    seasons that we've been involved with TNT.</p>

    <p>From 2008 through early 2015 we  continued to volunteer with TNT, as
    participants, mentors, and since 2010 I was a coach (check out my
    <a href="http://youtu.be/GMSKG8L6K78#t=2m13s">half-second</a> of fame in the
    info video) for TNT with an emphasis on training walkers for full- or
    half-marathons. This page provides a summary of the events that I've participated
    in in some capacity since we became involved with TNT. </p>

    <div id="event-map" name="event-map"></div>

    <h2>
        <span class="date">11/13/2015</span>
        Avengers Half Marathon
        <span class="time">3:17:55</span>
        (<a href="#event-map" onclick="recenter(map, eventPlaces[5].point, 12)">map</a>)
    </h2>

    <h2>
        <span class="date">1/11/2015</span>
        Disney World  Marathon (Goofy - Day 2)
        <span class="time">6:21:01</span>
        (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
    </h2>
    <p class="eventDescription">blah, blah, blah ...</p>

    <h2>
        <span class="date">1/10/2015</span>
        Disney World Half Marathon (Goofy - Day 1)
        <span class="time">2:45:55</span>
        (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
    </h2>
    <h2>
        <span class="date">10/19/2014</span>
        Duke City Half Marathon
        <span class="time">2:45:17</span>
```

```
57              (<a href="#event-map" onclick="recenter(map, eventPlaces[0].point, 11)">map</a>)
58          </h2>
59          <p class="eventDescription">blah, blah, blah ...</p>
60          <h2>
61              <span class="date">2/23/2014</span>
62              Princess Half Marathon
63              <span class="time">3:07:11</span>
64              (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
65          </h2>
66          <h2>
67              <span class="date">2/22/2014</span>
68              Princess Enchanted 10k
69              <span class="time">1:42:43</span>
70              (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
71          </h2>
72          <h2>
73              <span class="date">9/1/2013</span>
74              Disneyland Half Marathon
75              <span class="time">2:56:57</span>
76              (<a href="#event-map" onclick="recenter(map, eventPlaces[5].point, 12)">map</a>)
77          </h2>
78          <p class="eventDescription">blah, blah, blah ...</p>
79
80          <h2>
81              <span class="date">1/13/2013</span>
82              Disney World  Marathon (Goofy - Day 2)
83              <span class="time">6:46:57</span>
84              (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
85          </h2>
86          <p class="eventDescription">blah, blah, blah ...</p>
87
88          <h2>
89              <span class="date">1/12/2013</span>
90              Disney World Half Marathon (Goofy - Day 1)
91              <span class="time">3:22:48</span>
92              (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
93          </h2>
94          <p class="eventDescription">blah, blah, blah ...</p>
95
96          <h2>
97              <span class="date">9/29/2012</span>
98              Hot Chocolate 15k
99              <span class="time">1:56:46</span>
100             (<a href="#event-map" onclick="recenter(map, eventPlaces[6].point, 10)">map</a>)
101         </h2>
102         <p class="eventDescription">blah, blah, blah ...</p>
103
104         <h2>
105             <span class="date">6/9/2012</span>
106             Animas Valley/Steamworks Half Marathon
107             <span class="time">no time: coached</span>
108             (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
109         </h2>
110         <p class="eventDescription">blah, blah, blah ...</p>
```

```
111
112         <h2>
113             <span class="date">1/9/2012</span>
114             Disney World  Marathon (Goofy - Day 2)
115             <span class="time">6:56:28</span>
116             (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
117         </h2>
118         <p class="eventDescription">blah, blah, blah ...</p>
119
120         <h2>
121             <span class="date">1/8/2011</span>
122             Disney World Half Marathon (Goofy - Day 1)
123             <span class="time">3:29:00</span>
124             (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
125         </h2>
126         <p class="eventDescription">blah, blah, blah ...</p>
127
128         <h2>
129             <span class="date">6/19/2010</span>
130             Animas Valley/Steamworks Half Marathon
131             <span class="time">no time: coached</span>
132             (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
133         </h2>
134         <p class="eventDescription">blah, blah, blah ...</p>
135
136         <h2>
137             <span class="date">6/6/2010</span>
138             San Diego Rock 'n' Roll Marathon
139             <span class="time">no time: coached</span>
140             (<a href="#event-map" onclick="recenter(map, eventPlaces[2].point, 11)">map</a>)
141         </h2>
142         <p class="eventDescription">blah, blah, blah ...</p>
143
144         <h2>
145             <span class="date">10/18/09</span>
146             Nike Women's Marathon
147             <span class="time">7:13:05</span>
148             (<a href="#event-map" onclick="recenter(map, eventPlaces[3].point, 12)">map</a>)
149         </h2>
150         <p class="eventDescription">blah, blah, blah ...</p>
151
152         <h2>
153             <span class="date">9/6/2009</span>
154             Disneyland Half Marathon
155             <span class="time">3:43:05</span>
156             (<a href="#event-map" onclick="recenter(map, eventPlaces[5].point, 12)">map</a>)
157         </h2>
158         <p class="eventDescription">blah, blah, blah ...</p>
159
160         <h2>
161             <span class="date">1/11/2009</span>
162             Disney World Marathon
163             <span class="time">6:57:42</span>
164             (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
```

```
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">10/19/2008</span>
            Duke City Half Marathon
            <span class="time">3:09:42</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[0].point, 11)">map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

        <h2>
            <span class="date">6/21/2008</span>
            Animas Valley/Steamworks Half Marathon
            <span class="time">3:14:52</span>
            (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
        </h2>
        <p class="eventDescription">blah, blah, blah ...</p>

    </body>

    <!-- Let's put our JavaScript down here ----------------------------------------->
        <!-- Load the external JavaScript file with the map definition code -->
        <script src="js/mapPage_events.js"></script>

        <!-- Load the API in asynchronous mode and execute the initialize function
             when done -->
        <!-- The optional 'key-<API Key>' parameter is not used here but should
            be for a production map -->
        <script async defer
            src="https://maps.googleapis.com/maps/api/js?callback=initialize">
        </script>
    </body>
</html>
```

gmaps_events.js

```
var map;
var eventPlaces;

function initialize() {
    // Define a set of global coordinates for use throughout the web site
    // Place coordinates derived from GNIS database: http://geonames.usgs.gov/pls/gnispublic
    eventPlaces = [
        {
            name: "Albuquerque",
            point: new google.maps.LatLng(35.0889356,-106.5747462),
            label: "Albuquerque: Duke City Half Marathon"
        },
        {
            name: "Durango",
            point: new google.maps.LatLng(37.2752800,-107.8800667),
            label: "Durango: Animas Valley/Steamworks Half Marathon"
        },
```

```
18          {
19              name: "San Diego",
20              point: new google.maps.LatLng(32.7153292,-117.1572551),
21              label: "San Diego: San Diego Rock 'n' Roll Marathon"
22          },
23          {
24              name: "San Francisco",
25              point: new google.maps.LatLng(37.7749295,-122.4194155),
26              label: "San Francisco: Nike Women's Marathon"
27          },
28          {
29              name: "Orlando",
30              point: new google.maps.LatLng(28.5383355,-81.3792365),
31              label: "Orlando: Walt Disney World half- and full-marathon"
32          },
33          {
34              name: "Anaheim",
35              point: new google.maps.LatLng(33.8352932,-117.9145036),
36              label: "Anaheim: Disneyland Half Marathon"
37          },
38          {
39              name: "Albuquerque",
40              point: new google.maps.LatLng(35.0889356,-106.5747462),
41              label: "Hot Chocolate 15k"
42          }
43
44      ];
45
46      var myOptions = {
47          zoom: 4,
48          center: eventPlaces[0].point,
49          mapTypeId: google.maps.MapTypeId.ROADMAP};
50
51      map = new google.maps.Map(
52          document.getElementById("event-map"),
53          myOptions);
54
55      addMarkers(map,eventPlaces)
56  }
57
58  function recenter(mapName, latlon, zoomLevel) {
59      mapName.setCenter(latlon);
60      mapName.setZoom(zoomLevel)
61  }
62
63  function addMarkers(mapName, markerArray) {
64      for (index = 0; index < markerArray.length; index++) {
65          myMarker = new google.maps.Marker({
66              position: markerArray[index].point,
67              title: markerArray[index].label
68          });
69          myMarker.setMap(mapName)
70      }
71  }
```

Chapter 5

Week 5 - Module 3 - GIS and Services Oriented Architectures

Overview

- Geographic Information Systems
  - Data Types
  - Coordinate Systems
- Services Oriented Architectures
  - Historic Context
  - Current Model - Network Computing
  - Components
  - Interoperability Standards

Geographic Information Systems

Data Types - Vector

- Vector data represent phenomena that are associated with specific bounded locations, typically represented by:
  - Points
  - Lines
  - Polygons
- Vector data include:
  - The geometries that describe the area being referenced, and
  - Attributes associated with that area

For example, a census vector data product might include the geometries that define census tracts and attributes associated with each geometry: population, income, etc.

Data Types - Raster

- Raster data are frequently used to represent values for phenomena that vary continuously across space (e.g. elevation, concentration of air pollutants, depth to ground water, etc.)
- These values are encoded over a regular grid of observation locations with a specified grid spacing - often referred to as the spatial resolution of the dataset (i.e. 10m resolution for a standard USGS Digital Elevation Model product)
- Often parts of data collections that are repeated (i.e. remote sensing data products)

Accessing and Processing Raster and Vector Data

- ArcGIS - ArcCatalog
- QGIS - Dataset properties available through the "Metadata" tab
- Through metadata files available from the provider web site or embedded in the downloaded file

Accessing and Processing Raster and Vector Data - Programmatically

- Two geospatial libraries and their related utility programs provide information about and tools for modifying vector and raster data sets

OGR vector data access and information
GDAL raster data access and information

These libraries are the data access and processing foundation for a growing number of open source and commercial mapping systems

Information and documentation: GDAL Home Page | OGR Home Page

Coordinate Systems/Projections

- To convert locations from a 3-dimensional oblate spherical coordinate system (such as is commonly used to represent the surface of the earth) to a 2-dimensional representation in a map, a coordinate transformation must be performed.
- There are a limitless number of potential coordinate transformations possible, and a large number have been named and defined that meet specific cartographic or other requirements

EPSG Codes

- A catalog of numeric codes and associated coordinate transformation parameters is maintained by the International Association of Oil & Gas Producers (OGP) - the successor scientific organization to the European Petroleum Survey Group (EPSG)
- These numeric codes are used by many desktop and online mapping systems to document and represent the coordinate systems of available data and services
- Links to an online version of the registry and downloadable databases of the registry are available from: http://www.epsg.org/Geodetic.html.

Projection Parameters

The parameters that define a map projection may be looked up in a number of online locations:

EPSG registry Helpful if you already know the EPSG code of the projection you are looking for - http://www.epsg-registry.org/

GeoTIFF Projection List Helpful if you know the name of one of the broadly used projections - uneven performance of links - http://www.remotesensing.org/geotiff/proj_list/

SpatialReference.org Decent search tool, includes non-EPSG as well as EPSG projection information, multiple descriptions of projection parameters - http://spatialreference.org/

Services Oriented Architectures

Where have we come from - ENIAC (1946)

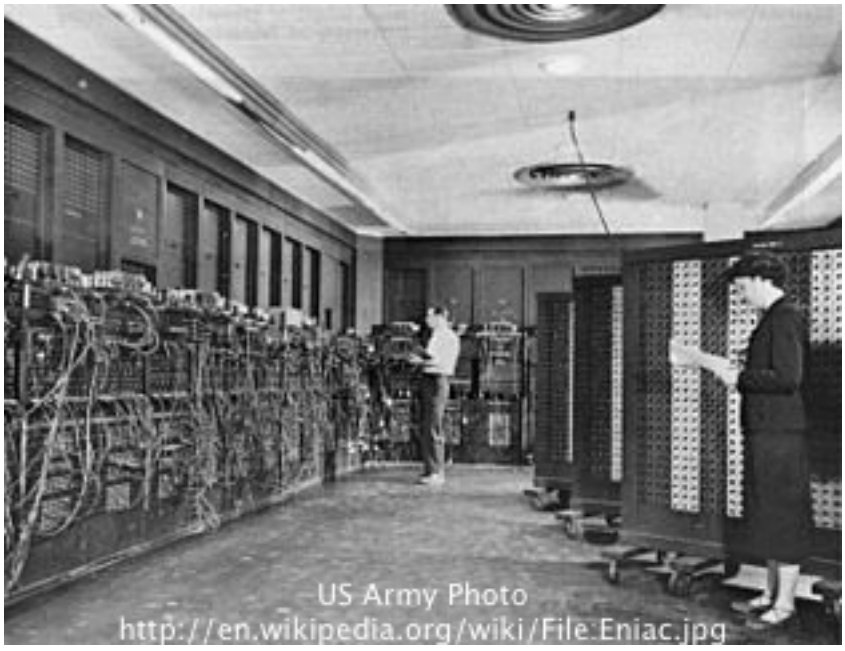


Figure 5.1: ENIAC Computer

- First general purpose electronic computer
- Programmable, but could not store programs

Where have we come from - Early Client-Server Computing (1960s)

- Mainframe computers to which client terminals connected over a local network
- Computing performed by server, client purely a display device

Where have we come from - Personal Computers (1970s)

- Desktop computers capable of running a variety of operating systems and applications
- In some environments can be interconnected to a central local server

Figure 5.2: IBM 704 Mainframe Computer

Figure 5.3: Model 33 ASR Teletype

Figure 5.4: TeleVideo 925 ASCII Terminal

Figure 5.5: IBM 5150 Personal Computer

Figure 5.6: Apple I Personal Computer

Now - Network computing

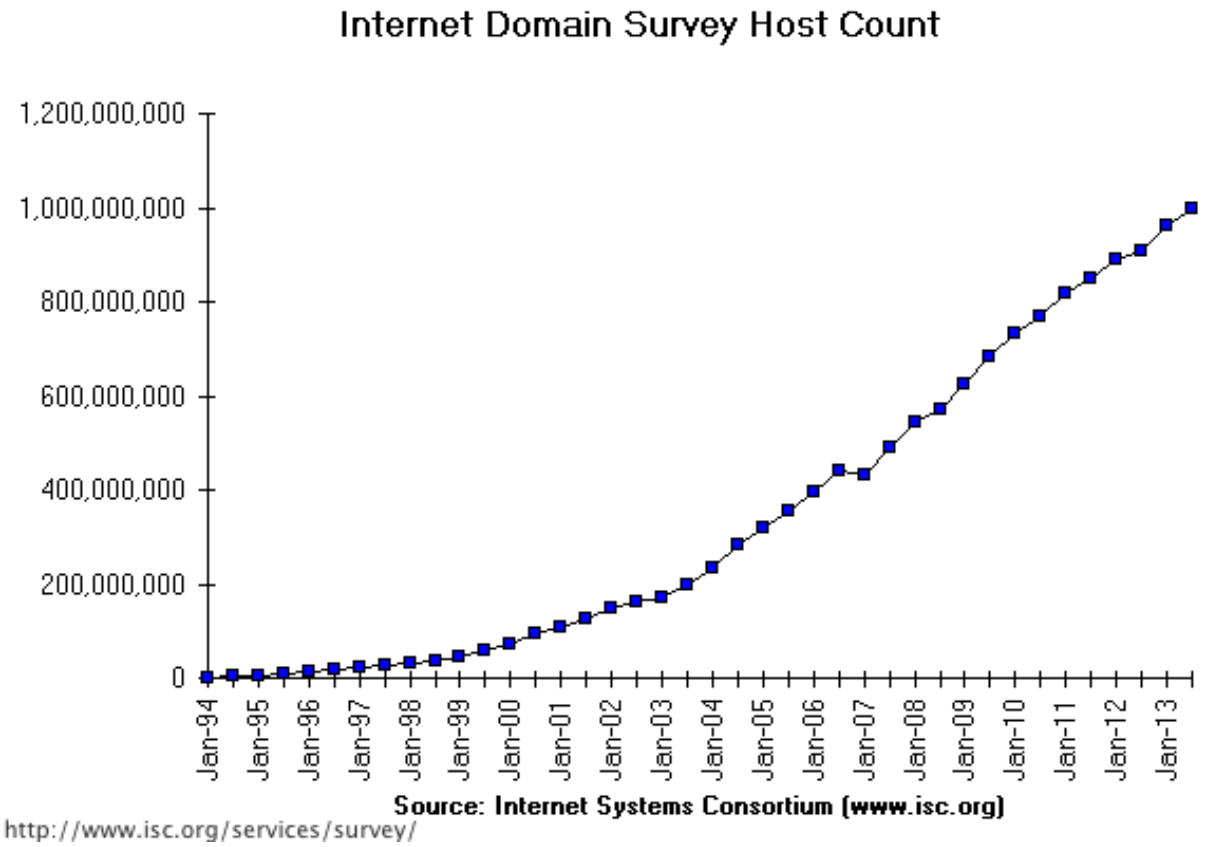


Figure 5.7: World Internet Hosts 1/94-1/13. Image courtesy IWS - http://www.isc.org/services/survey/

Network Computing Timeline

- Predecessor to the Internet - ARPANET (1969). Interconnection between UCLA and SRI (Menlo Park)
- Adoption of TCP/IP as next generation protocol for ARPANET (1983)
- NSF commissions construction of NSFNET, also based upon TCP/IP (1983)
- NSFNET opened to commercial connections (1988). Led to interconnection of multiple, previously separate networks into an "Internet"
- Growth of internet users has expanded rapidly over the past decade

In a Phrase ...

The current networking computing model consists of *Components Interacting* with Each Other

So - We Need to Answer the Following Questions

What are components?

What does it mean to interact?

The Big Picture - Services Oriented Architectures

- Services Oriented Architecture (SOA) for Geospatial Data and Processing
  - Data, Processing & Client Tiers
- Open Geospatial Consortium Interoperability Standards
  - WMS, WFS, WCS
- Geospatial Metadata Standards
  - ISO 19115, FGDC
- Internet Standards
  - Web: HTML, CSS, JavaScript, XML
  - SOAP - Simple Object Access Protocol
  - REST - Representation State Transformation

The Pieces - Components

Key Components - Data

Database systems

- Optimized for storing massive quantities of tabular data
- May be spatially enabled to support the storage of geometries (points, lines, polygons) in addition to related attribute data
- Standard language (Structured Query Language [SQL]) for interacting with many databases
- Broad support for accessing the contents of databases from many other applications and programming languages, for example:
  - Spreadsheets
  - Statistical Software
  - Geographic Information Systems (GIS)

Key Components - Data

File-based data

- Often stored on the file system
- Sometimes difficult represent data within a database structure (i.e. binary data)
- May be in a wide variety of formats
  - XML
  - ASCII Text (e.g. CSV, tab-delimited)
  - Binary files
  - Excel Spreadsheets
  - Word Processing Documents
  - Geospatial data (e.g. imagery)
- Remotely Accessible Data
  - Some data may be provided through reference to an external network resource (i.e. a web address, or other identifier) or service

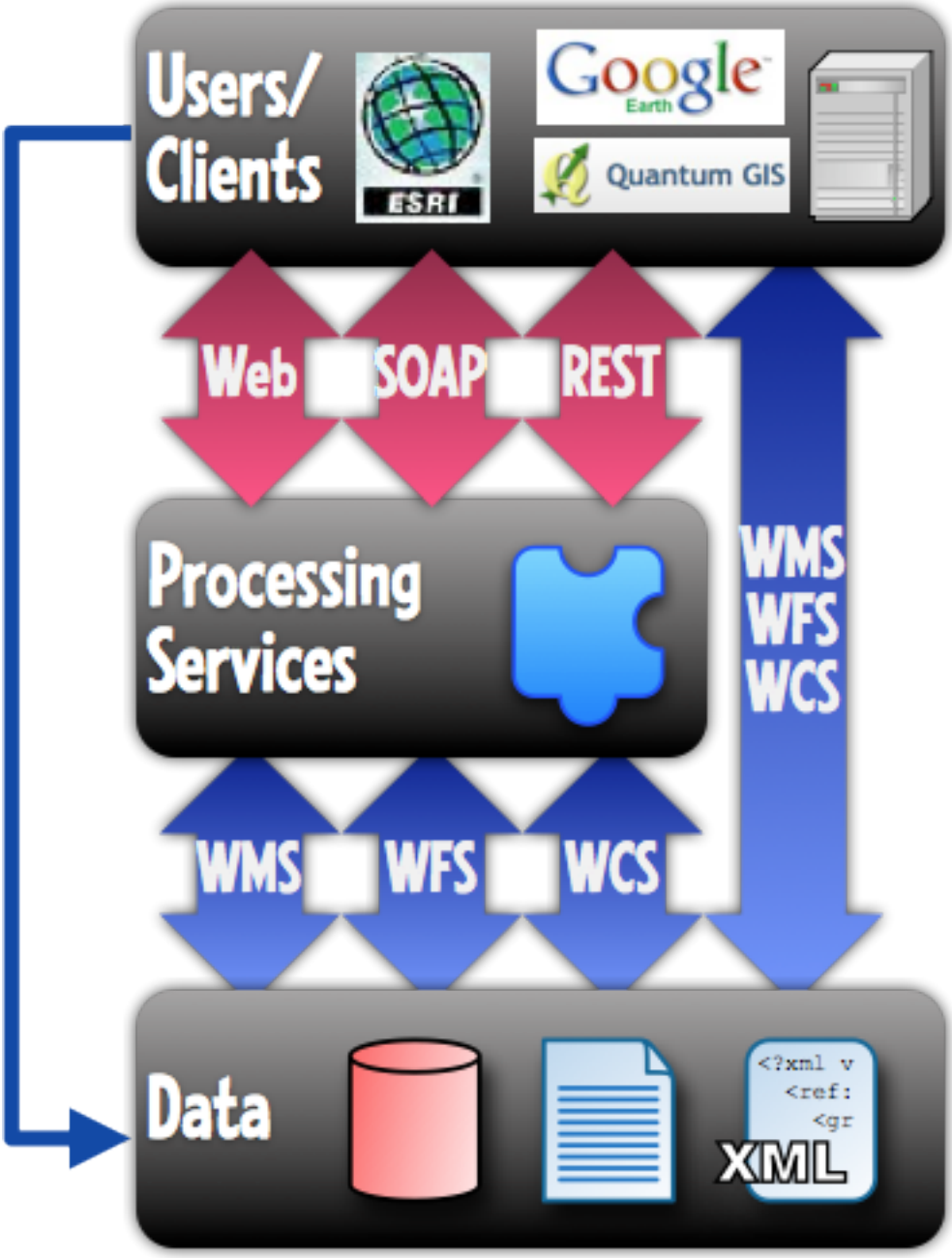


Figure 5.8: SOA Illustration

Key Components - Processing Services

- Perform modification of source data to generate a new data product
- May be "chained" together to create a processing "workflow". Output from one processing service may be used as the input to another
- May be simple OGC services; or complex data processing, analysis, or visualization services. Examples include
  - Extraction of a subset of a large data set based upon provided search criteria
  - Generation of a map from a collection of data
  - Fusion of two data products into a single derived product (e.g. vegetation indices calculated from multiple remote sensing images)
  - Calculation of statistical information for an input product, and delivery of the statistical summary

Key Components - Clients

- Any system that accesses the services provided by the system may be considered a "client"
- That system may be manually operated by a human user, or triggered automatically by software
- Human operated clients include
  - Web-based applications
  - Desktop applications such as Geographic Information Systems and Statistical Analysis tools
- Machine clients include
  - Data processing services that translate requests to them into requests for other system services
  - Regularly scheduled requests that are automatically triggered by external computer systems.

The Glue - Interoperability Standards / Service Interfaces

Open Geospatial Consortium Interoperability Standards

Open Geospatial Consortium (OGC) Standards

- Two Classes of Standards Considered Here
  - Geospatial Product Access Standards
  - Geospatial Data and Representation Standards
- Product Access Standards
  - Web Map Services (WMS)
  - Web Feature Services (WFS)
  - Web Coverage Services (WCS)
- Data and Representation Standards
  - Geography Markup Language (GML)
  - KML (formerly known as Keyhole Markup Language)

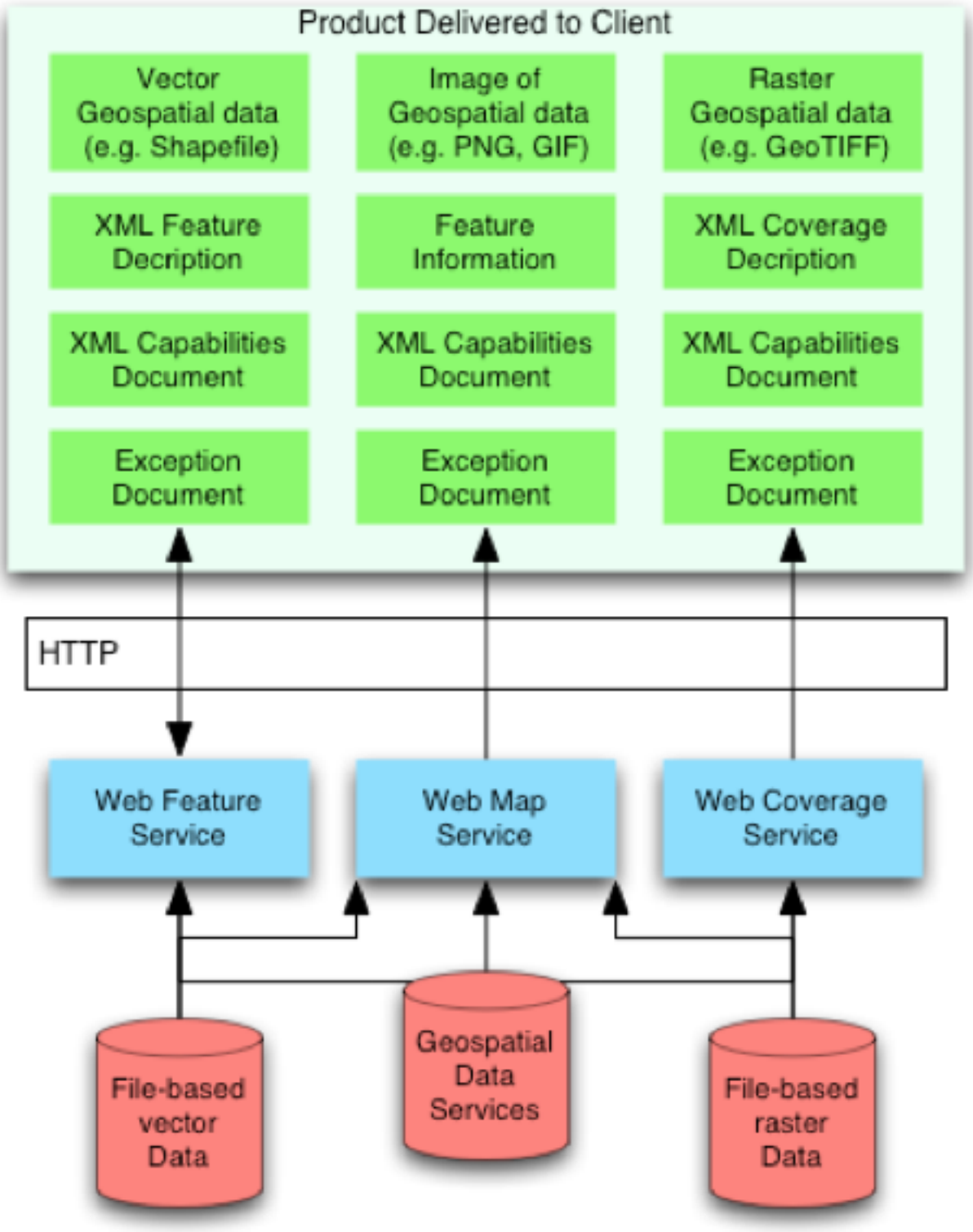


Figure 5.9: Comparison of OGC Service Models

Figure 5.10: WMS request result for Bernalillo County Landsat Mosaic from NM RGIS link

Comparison of OGC Service Models

OGC Web Map Services (WMS)

```
http://gstore.unm.edu/apps/rgis/datasets/
b030ab7b-86e3-4c30-91c0-f427303d5c77/
services/ogc/wms?
    VERSION=1.1.1&&
    SERVICE=WMS&
    REQUEST=GetMap&
    SRS=EPSG:4326&
    FORMAT=image/jpeg&
    STYLES=&
    LAYERS=bernalillo_tm2011&
    TRANSPARENT=TRUE&
    WIDTH=521&
    HEIGHT=200&
    bbox=-107.207,34.8404,-106.143,35.2487
```

OGC Web Feature Services (WMS) Characteristics

- HTTP GET (required), HTTP POST (optional)
- Requests:
  - `GetCapabilities`
  - `GetMap`
  - `GetFeatureInfo`
- Returns
  - Mapped data
  - XML Capabilities Document, Feature Attributes
- Includes support for time-based requests

OGC Web Feature Services (WFS) Characteristics

- Either HTTP GET or POST required
- Requests
  - `GetCapabilities`
  - `DescribeFeatureType`
  - `GetFeature/GetFeatureWithLock`
  - `GetGmlObject`
  - `LockFeature`
  - `Transaction`
- Returns
  - XML (GML)
  - Capabilities
  - Feature Data

OGC Web Coverage Services (WCS) Characteristics

- Either HTTP GET or POST required
- Requests
  - `GetCapabilities`
  - `DescribeCoverage`
  - `GetCoverage`
- Returns
  - Geospatial data for coverage
  - XML Capabilities
- Includes support for time-based requests

OGC Geography Markup Language (GML)

- GML is an XML grammar for representing geospatial features and their associated attributes
- In its generic form it can encode points, lines, and polygons and their associated attributes
- As an XML schema GML was designed to be extensible by communities of practice for consistent encoding of geographic data more richly than allowed by the generic default model
- GML documents representing large complex geometries can be quite large - therefore slow to transfer over the Internet

OGC KML

- An XML specification that supports the encoding of representation and embedding of geospatial data for use in geospatial viewers
- Began as the underlying representation language of Google Earth (originally developed by Keyhole for their virtual Earth viewer)
- Adopted as an OGC standard in 2008
- Supports data linkage through
  - Embedding
  - Reference through external URLs - with WMS specifically supported through *parameterization*
- Includes support for the representation of time in relation to data objects

Implementation of the OGC Standards

- WMS
  - 1.3.0 - 351 implementations
  - 1.1.1 - 524
  - 1.1 - 257
  - 1.0 - 293
- WFS
  - 2.0 - 62
  - 2.0 transactional - 14
  - 1.1.0 - 280
  - 1.1.0 transactional - 76
  - 1.0.0 - 340
  - 1.0.0 transactional - 127
- WCS
  - 2.0 - Core - 7
  - 1.1.2 - 22
  - 1.1.1 Corregendum 1- 56
  - 1.1.0 - 30
  - 1.0.0 Corregendum - 210

Implementation information based upon OGC Implementation Statistics - Accessed 2/2016

Implementation of the OGC Standards

- KML
  - 2.2.0 - 106
  - 2.2 Reference (Best Practice) - 11
  - 2.1 Reference (Best Practice) - 77
- GML
  - 3.3 - 6
  - 3.2.1 - 140
  - 3.1.1 - 161
  - 3.0 - 145
  - 2.1.2 - 168
  - 2.1.1 - 117
  - 2.0 - 82
  - 1.0 - 20

Implementation information based upon OGC Implementation Statistics - Accessed 2/202016

OGC Summary

The OGC web service specifications support key geospatial data access requirements

WMS visualization of geospatial data through simple web requests
WFS delivery of geospatial data (typically points, lines, and polygons) in a format that is usable in GIS and other applications
WCS delivery of geospatial data (typically, but not limited to, raster data) usable in other applications

OGC Summary

The OGC data and representation standards support data exchange and higher level representation

GML XML schema for the representation of features and associated attributes. It may be extended for use by specific communities of users (i.e. ecological data models)

KML XML schema that supports the combination of embedded data and external data into a complete representation model that may be used by client applications to present the data through a user interface (e.g. Google Earth, WorldWind)
