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>
```

```
        <!-- The head is where you include pointers to external resources
        (i.e. style sheets and javascript files), blocks of Javascript code
        , styles, etc.  -->
        <title>The page title also goes in here</title>
    </head>
    <body>
        <!-- The body is where you put all of the content for the page
        (i.e. the material that will be displayed in the web browser)  -->
        <h1>Headers</h1>
        <div>Generic blocks of content</div>
        <p>Paragraphs</p>
        <table>Tables</table>
        <img ...>Images</img>
        <form ...>Forms</form>
        <ul>Unordered Lists</ul>
        <ol>Ordered Lists</ol>
        <li>List Items</li>

        <!-- Javascript can go here as well -->
    </body>
</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>
```

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

link to example/Preview

More Complete Web Page Example

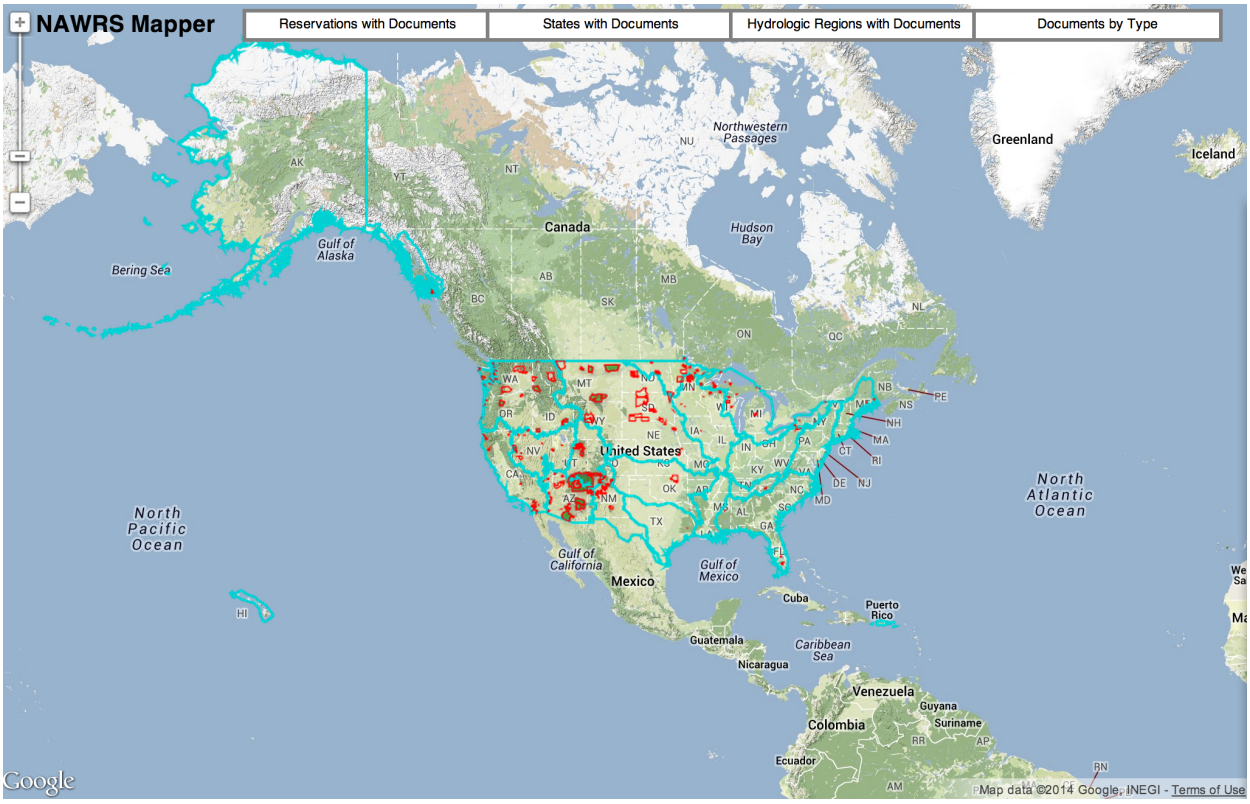


Figure 2.1: NAWRS Mapper. *HTML*: 39 Lines; *CSS*: 136 Lines; *core.js*: 515 Lines + Google Maps API and JQuery Framework

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

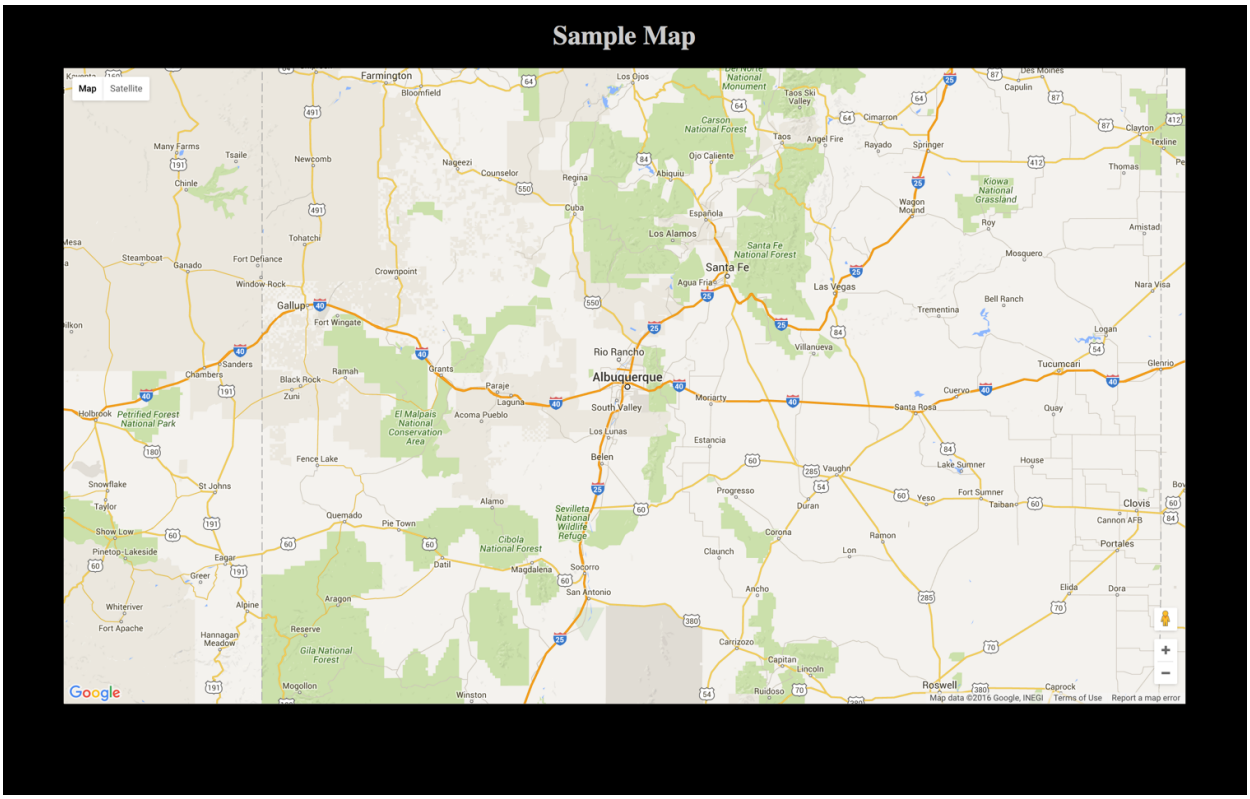
- Elevation
  - Delivery of elevation data for locations or paths
- Streetview
  - Integration of Google Streetview within a DOM element
- Maximum Zoom
  - Provides information about the maximum available zoom level

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>
```

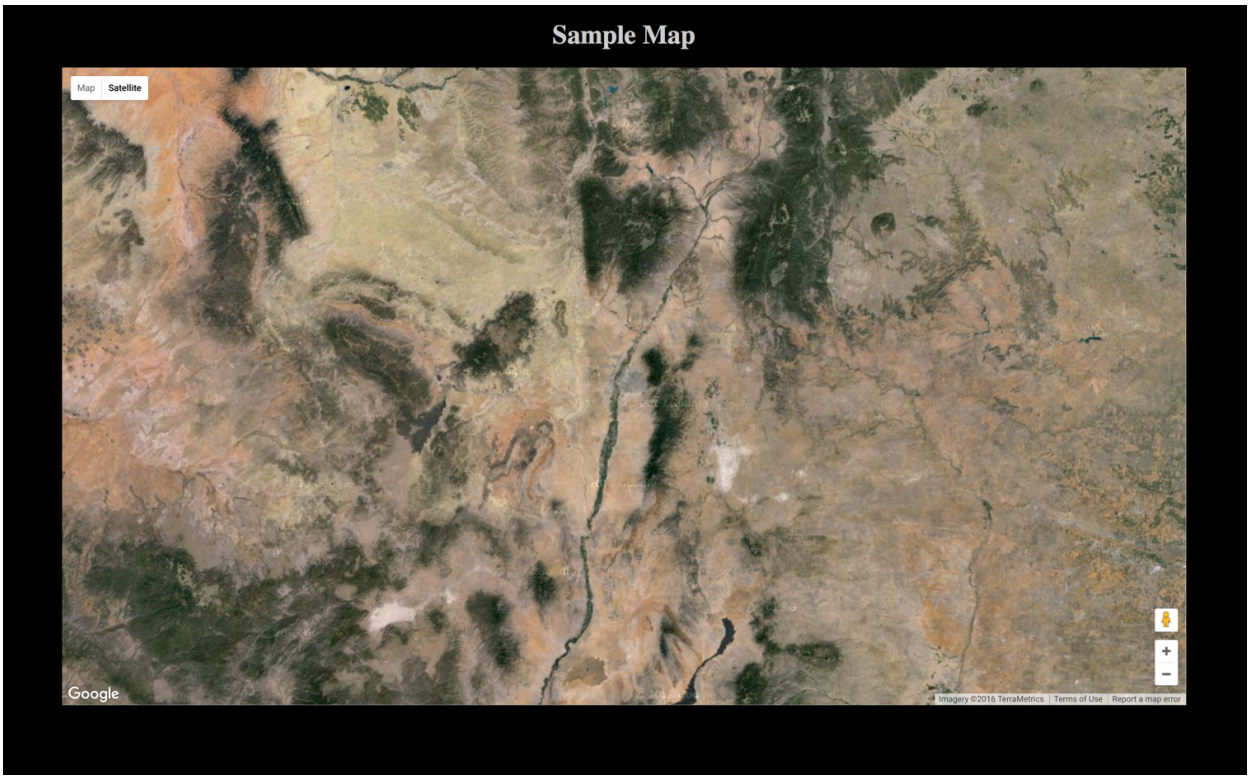
mapPage.css

```
/* 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

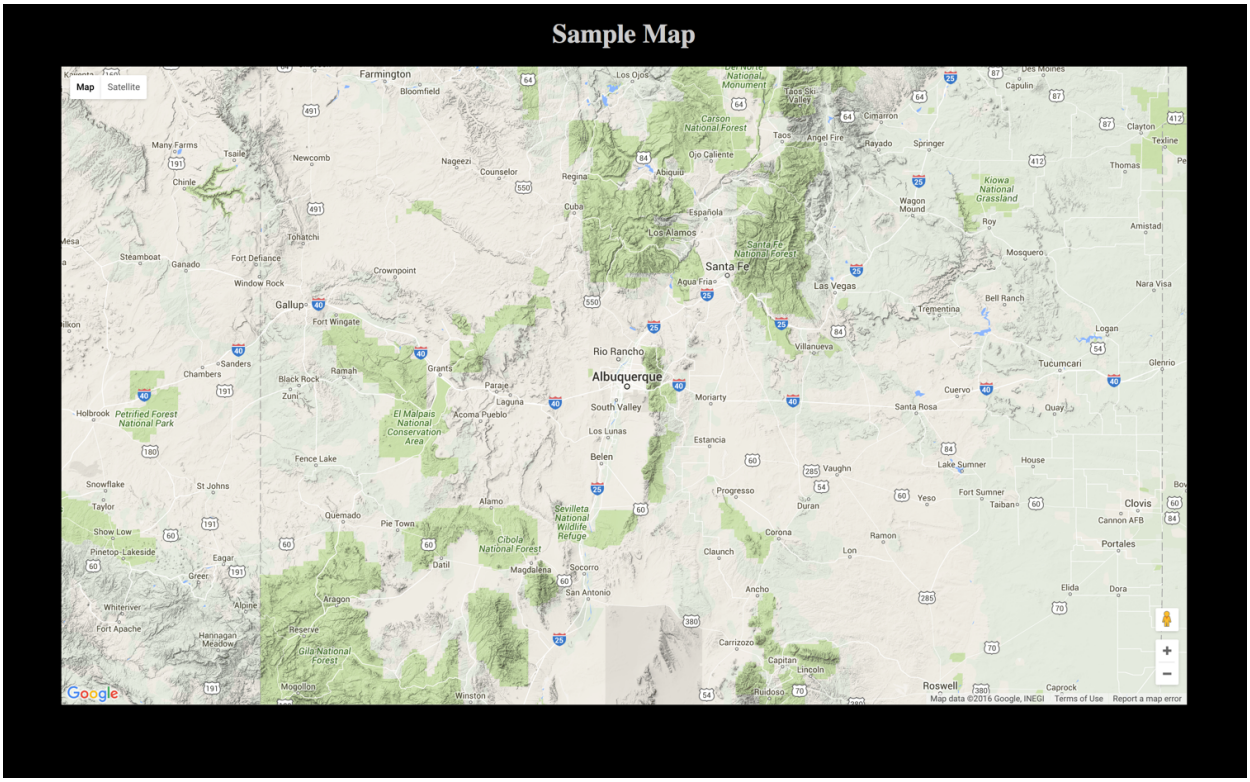
```
<!DOCTYPE html>
<html>
```

```
    <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>
```

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

mapPage_04.js

```
1   function initialize() {
2       var classroom = new google.maps.LatLng(35.084280,-106.624073)
3       var mapOptions = {
4           zoom: 8,
5           center: classroom,
6           mapTypeId: google.maps.MapTypeId.TERRAIN
7       };
8       var map = new google.maps.Map(
9           document.getElementById("map_canvas"),
10          mapOptions);
11  }
12
```

Simple - Hybrid - Zoomed

Simple - Hybrid - Zoomed Code

gmaps05.html

```
1   <!DOCTYPE html>
2   <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>

```

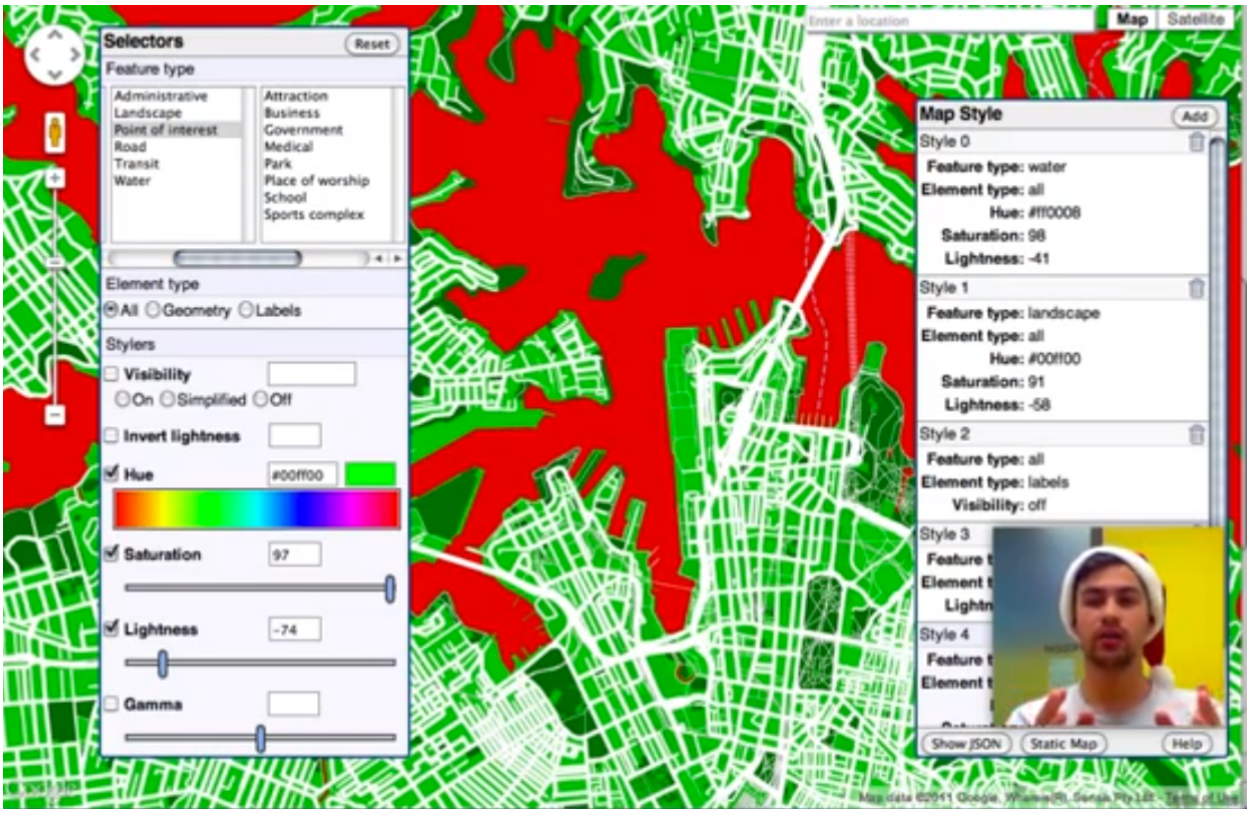


Figure 4.1: Google Maps Styled Maps Wizard link

```
        <!-- 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

Bringing It All Together

gmaps_events.html

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

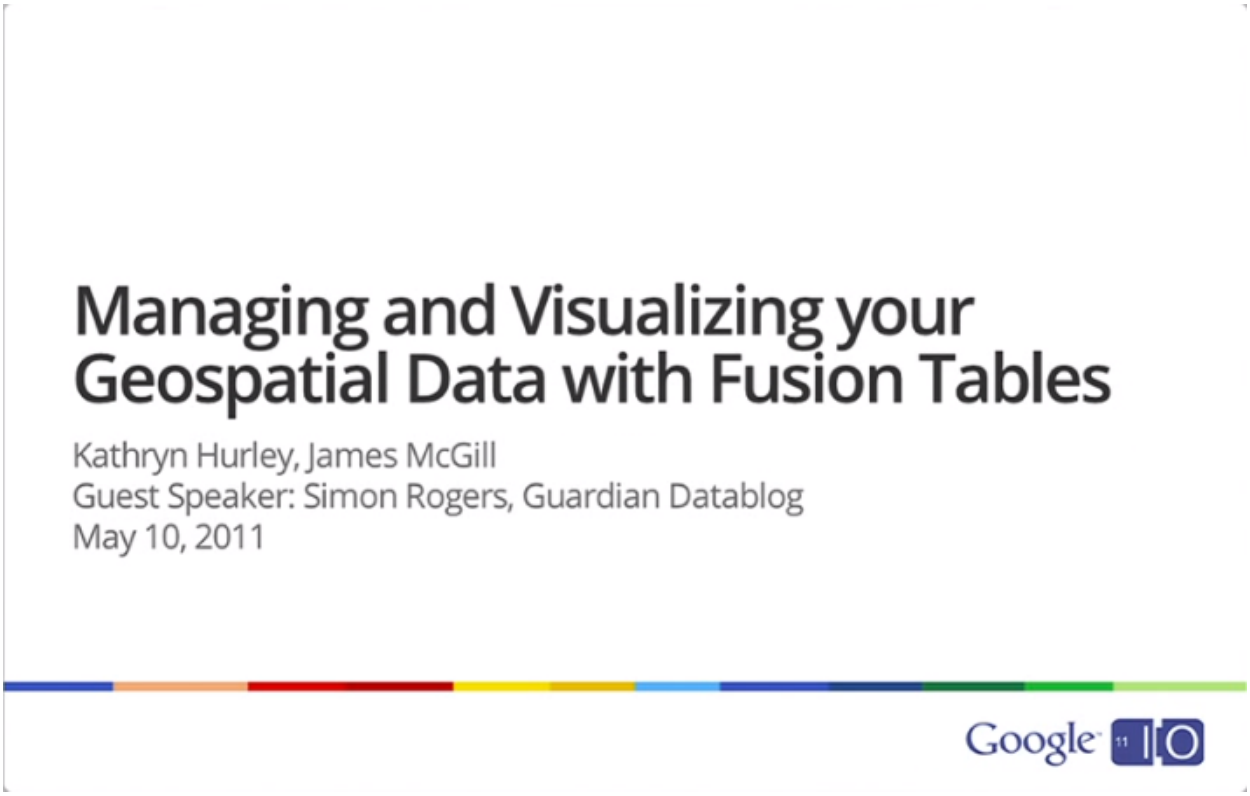


Figure 4.2: Google Fusion Tables Introduction Video link

```
<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>
```

```
29
30          <div id="event-map" name="event-map"></div>
31
32          <h2>
33              <span class="date">11/13/2015</span>
34              Avengers Half Marathon
35              <span class="time">3:17:55</span>
36              (<a href="#event-map" onclick="recenter(map, eventPlaces[5].point, 12)">map</a>)
37          </h2>
38
39          <h2>
40              <span class="date">1/11/2015</span>
41              Disney World  Marathon (Goofy - Day 2)
42              <span class="time">6:21:01</span>
43              (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
44          </h2>
45          <p class="eventDescription">blah, blah, blah ...</p>
46
47          <h2>
48              <span class="date">1/10/2015</span>
49              Disney World Half Marathon (Goofy - Day 1)
50              <span class="time">2:45:55</span>
51              (<a href="#event-map" onclick="recenter(map, eventPlaces[4].point, 10)">map</a>)
52          </h2>
53          <h2>
54              <span class="date">10/19/2014</span>
55              Duke City Half Marathon
56              <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>)
165         </h2>
166         <p class="eventDescription">blah, blah, blah ...</p>
167 
168         <h2>
169             <span class="date">10/19/2008</span>
170             Duke City Half Marathon
171             <span class="time">3:09:42</span>
172             (<a href="#event-map" onclick="recenter(map, eventPlaces[0].point, 11)">map</a>)
173         </h2>
174         <p class="eventDescription">blah, blah, blah ...</p>
175 
176         <h2>
177             <span class="date">6/21/2008</span>
178             Animas Valley/Steamworks Half Marathon
179             <span class="time">3:14:52</span>
180             (<a href="#event-map" onclick="recenter(map, eventPlaces[1].point, 10)">map</a>)
181         </h2>
182         <p class="eventDescription">blah, blah, blah ...</p>
183 
184     </body>
185 
186     <!-- Let's put our JavaScript down here ----------------------------------------->
187         <!-- Load the external JavaScript file with the map definition code -->
188         <script src="js/mapPage_events.js"></script>
189 
190         <!-- Load the API in asynchronous mode and execute the initialize function
```

```
191             when done -->
192         <!-- The optional 'key-<API Key>' parameter is not used here but should
193             be for a production map -->
194         <script async defer
195             src="https://maps.googleapis.com/maps/api/js?callback=initialize">
196         </script>
197     </body>
198 </html>
```

gmaps_events.js

```
1  var map;
2  var eventPlaces;
3
4  function initialize() {
5      // Define a set of global coordinates for use throughout the web site
6      // Place coordinates derived from GNIS database: http://geonames.usgs.gov/pls/gnispublic
7      eventPlaces = [
8          {
9              name: "Albuquerque",
10             point: new google.maps.LatLng(35.0889356,-106.5747462),
11             label: "Albuquerque: Duke City Half Marathon"
12         },
13         {
14             name: "Durango",
15             point: new google.maps.LatLng(37.2752800,-107.8800667),
16             label: "Durango: Animas Valley/Steamworks Half Marathon"
17         },
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)

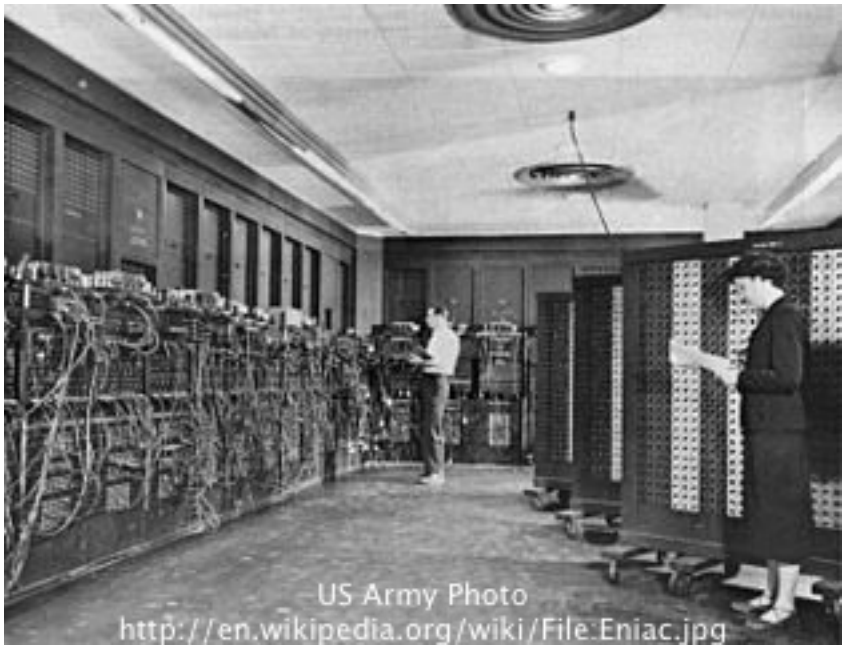


Figure 5.1: ENIAC Computer

- 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

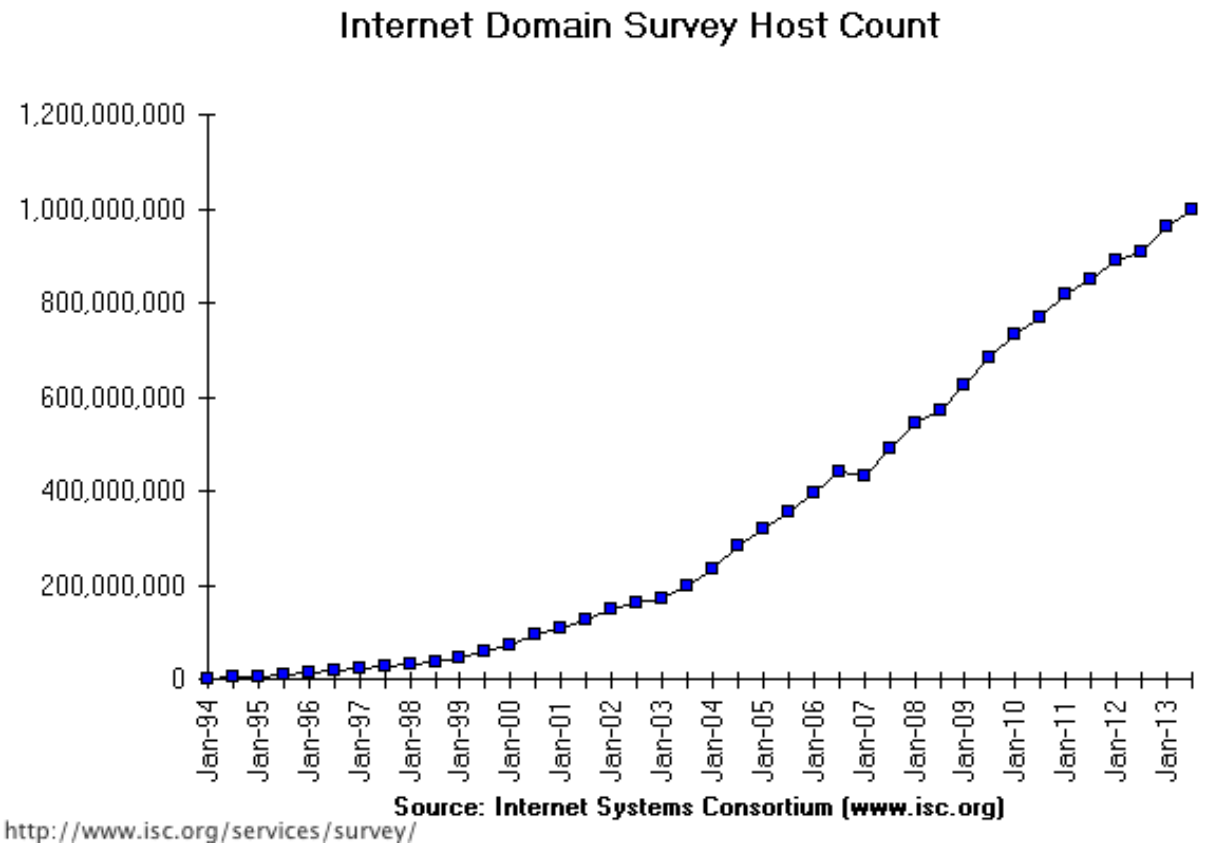


Figure 5.7: World Internet Hosts 1/94-1/13. Image courtesy IWS - http://www.isc.org/services/survey/

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

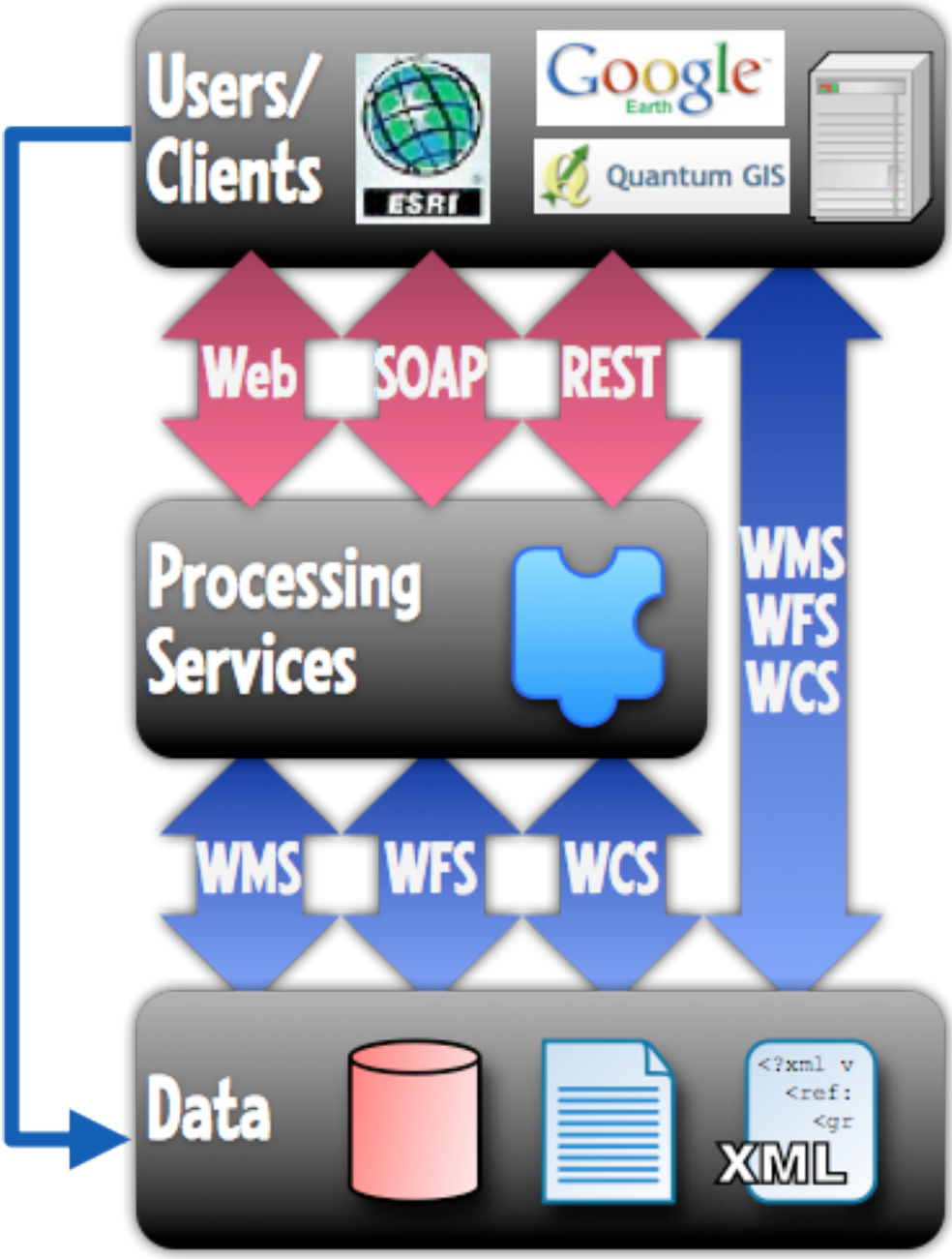


Figure 5.8: SOA Illustration

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)

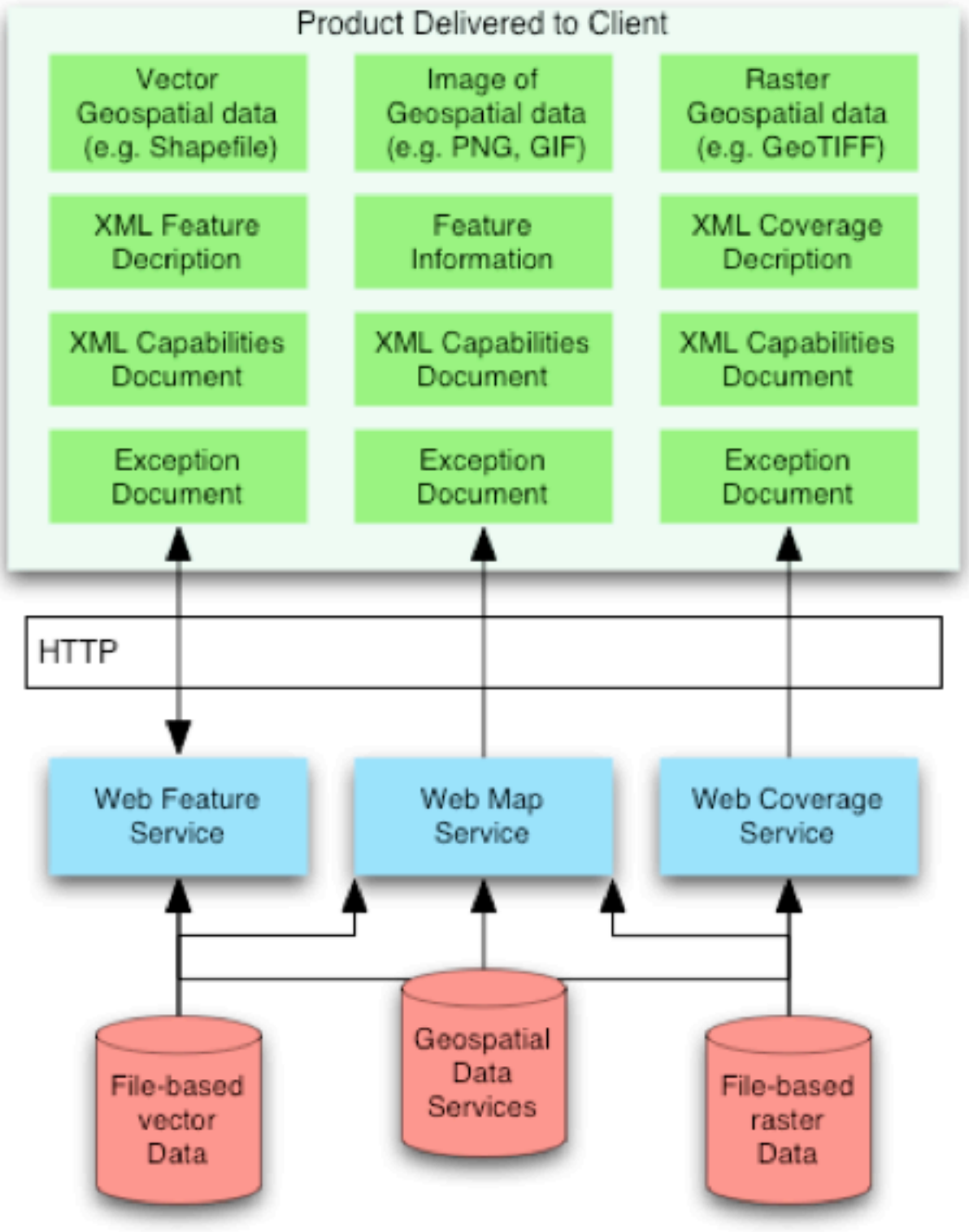


Figure 5.9: Comparison of OGC Service Models

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)

Chapter 6

Week 6 - Module 4a - Interoperability Standards. WMS, KML and XML

Overview

- Extensible Markup Language - XML
  - Definition of a markup language
  - Requirements
  - Extensible ???
- KML - AKA Keyhole Markup Language
  - An XML Document Format
  - Combined representation of spatial data and time
- OGC Web Map Services (WMS)
  - Requests and Results
  - GetCapabilities, GetMap, GetFeatureInfo
- Integration of WMS into KML

Extensible Markup Language - XML

XML Background

- Defined as a markup language profile of Standard Generalized Markup Language (SGML - ISO 8879:1986)
- XML 1.0 released as a W3C Recommendation in 1998
  - currently in 5th edition, released in 2008
  - version 1.1 released in 2004, but is not recommended for use unless the "new characters in XML names, new line-end conventions, and references to control characters enabled with XML version 1.1 are needed".

XML Design Goals

- XML shall be straightforwardly usable over the Internet.

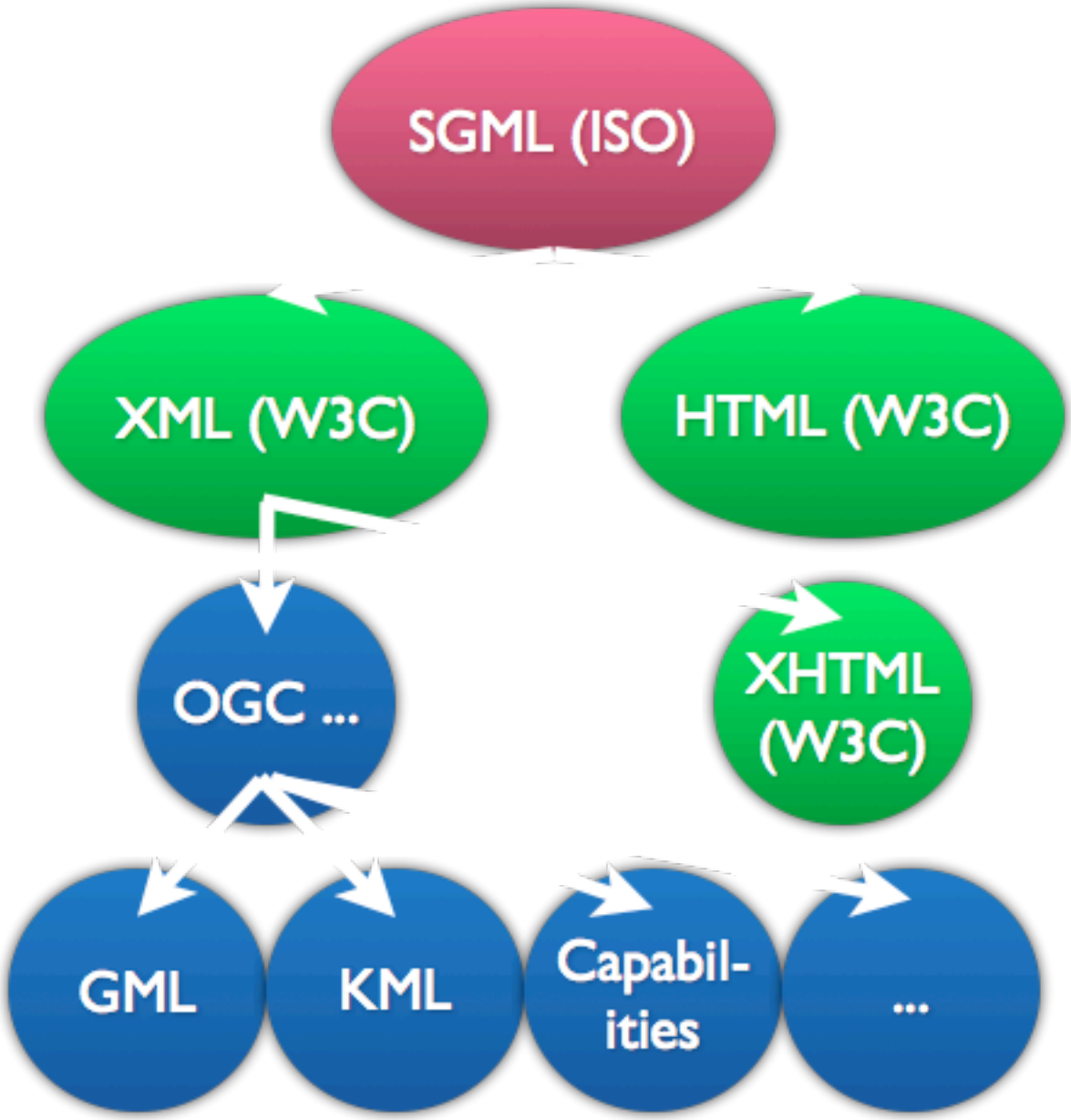


Figure 6.1: SGML Relationship with XML and HTML

- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

From XML 1.0 (5th ed.) Recommendation

XML Structure - Well Formed / Valid

- Well Formed XML - a document that conforms to the structural definition of XML. Either well-formed, or not XML
- Valid XML - a document that is both well-formed and conforms to a specific content structure defined by
  - A Document Type Definition (DTD) - the original XML specification for the definition of the content of a specific XML document
  - A Schema document - defined in a variety of languages (e.g. W3C Schema, RELAX NG, Schematron, ISO DSDL, etc.)

XML Wikipedia Article

Simple XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Some comment would go here to describe this document ... -->
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
</note>
```

XML Source (modified from original): w3schools

XML Prolog

Includes XML Declaration and Comment

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Some comment would go here to describe this document ... -->
```

XML Elements

Define blocks of content

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
</note>
```

XML Root Element

- Required
- There is only one
- It must be a pair of opening and closing tags

```
<note>
    ...
    ...
    ...
    ...
</note>
```

XML Content Elements

- Contain all other document content
- May be paired opening and closing tags, *or*
- May be self-closing with a terminal "/" in the element, e.g. `<br />`

```
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
```

XML Attributes

Define additional information about elements as *name=value* pairs.

```
    <body type="instruction" >Don't forget me this weekend!</body>
```

XML Element Content

The material contained between the opening and closing tags of an *Element*.

```
    <body type="instruction" >Don't forget me this weekend!</body>
```

Valid XML?

Why is this XML *well-formed* but not *valid*?

There is no DTD or Schema defined for the document against which it can be validated

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body type="instruction" >Don't forget me this weekend!</body>
</note>
```

Common XML Constructs

Document Type Declaration (DTD) references (PROLOG) definition, either by reference or by direct inclusion, the allowed structure of an XML document, for example:

```
<!DOCTYPE greeting SYSTEM "hello.dtd">
```

CDATA Sections blocks of XML that contain characters that would otherwise be recognized as XML markup, for example:

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

XML Namespace Declarations additional information included in elements to distinguish between duplicate element names, for example (declared in lines 1-3, used in lines 5-17):

```
<root
    xmlns:h="http://www.w3.org/TR/html4/"
    xmlns:f="http://www.w3schools.com/furniture">

<h:table>
    <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
    </h:tr>
</h:table>
<f:table>
    <f:legs>4</f:legs>
    <f:cost>300</f:cost>
    <f:width>3</f:width>
    <f:length>5</f:length>
    <f:height>4</f:height>
</f:table>
</root>
```

KML

KML Background

- An XML grammar originally developed as Keyhole Markup Language by Keyhole, Inc. for use in their Keyhole Earth Viewer.
- Google acquired Keyhole, Inc. in 2004
- KML version 2.2 became an OGC standard in 2008
- Two delivered KML file formats

KML an XML document, with a ".kml" extension that is directly readable and editable

KMZ a compressed (zipped) file with a ".kmz" extension[1], that contains at least a KML document, but may contain other files as well.

KML Capabilities

- Annotate the Earth
- Specify icons and labels to identify locations on the surface of the planet
- Create different camera positions to define unique views for KML features
- Define image overlays to attach to the ground or screen
- Define styles to specify KML feature appearance
- Write HTML descriptions of KML features, including hyperlinks and embedded images
- Organize KML features into hierarchies using `folder` elements
- Locate and update retrieved KML documents from local or remote network locations
- Define the location and orientation of textured 3D objects

KML Content

- Model for encoding 2- and 3-dimensional geometries for use in 2-D mappers and 3-D virtual globe applications
- Uses latitude-longitude (based upon WGS84 datum) for encoding horizontal position
- Represents altitude in Meters (based upon the WGS84 ellipsoid and EGM96 geoid)

2D and 3D KML Sample

```
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
    <Placemark>
        <Polygon>
            <altitudeMode>
                clampToGround
            </altitudeMode>
            <outerBoundaryIs>
                <LinearRing>
                    <coordinates>
                        -135,78.5,300000
                        -135,12.5,300000
                        -45,12.5,300000
```

[1] A `KMZ` file may be extracted and its contents examined by many zipfile utilities if you replace the `.kmz` extension with `.zip` prior to trying to extract

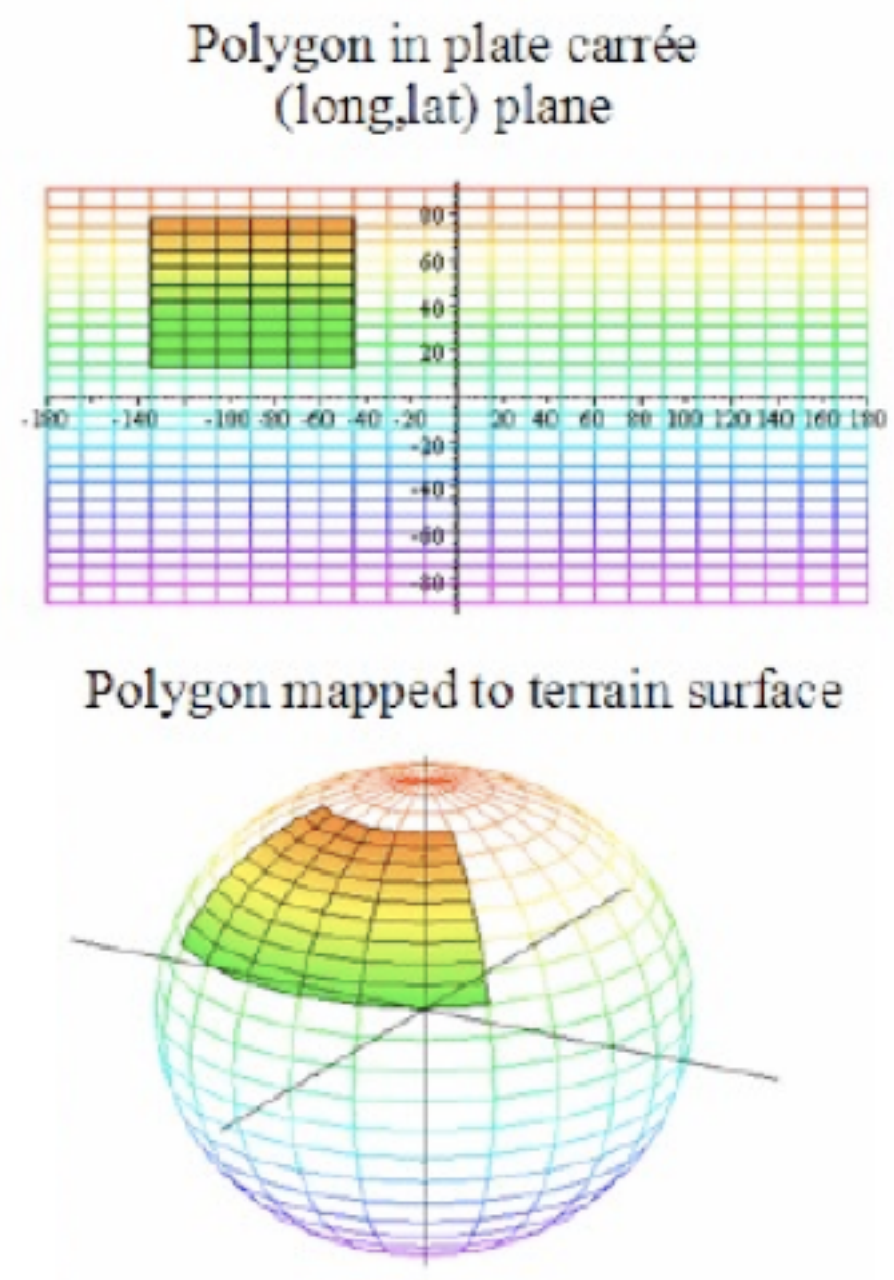


Figure 6.2: Illustration of polygon in both planar and terrain surface

```
                        -45,78.5,300000
                        -135,78.5,300000
                    </coordinates>
                </LinearRing>
            </outerBoundaryIs>
        </Polygon>
    </Placemark>
</Document>
</kml>
```

KML Example

Example from: KML 2.2 Specification (fig. 6, pg. 21)

High-Level KML Content Types

Features including documents, folders, placemarks, network links
Geometries including points, linestrings, polygons, models, locations
Overlays including ground overlays, lat-lon boxes, photo overlays, screen overlays
Styles styles, substyles, icons, label styles
Links read, update, create, delete, change
Views camera, look at
Time time span, timestamp

KML Demonstration and References

New Mexico State Boundary KML File | KMZ File (from NM RGIS)

Google Code KML Documentation

OGC KML Implementation specification

OGC Web map Services - WMS

WMS - Overview

- Open Geospatial Consortium standard for requesting
  - Service Metadata (`GetCapabilities`) - an XML file representing information about a specific WMS service and its component layers
  - Map Images (`GetMap`) - graphic files representing one or more layers from a single WMS service for a specified area of interest, and, optionally, for a specified point in time
  - Feature Information (`GetFeatureInfo`) - a basic representation (in a variety of formats) of the attributes associated with a specific pixel location in a map image
- A WMS will return to the requesting system one of the above products OR an error message (in XML by default)
- Related Style Layer Descriptor standard supports dynamic updating of visualization options
- OGC WMS Documentation Access Page

WMS *GetCapabilities* Request

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
WMTVER = 1.0.0	R				Request version
VERSION = version		O	O	O	Request version
SERVICE = WMS	R	R	R	R	Service type
REQUEST = capabilities	R				Request name
REQUEST = GetCapabilities		R	R	R	Request name
UPDATESEQUENCE = string		O	O	O	Sequence number or string for cache control
Vendor-specific parameters	O				Vendor-specific parameters

R=Required / O=Optional

WMS *GetMap* Request (Core)

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
WMTVER = 1.0.0	R				Request version
VERSION = version		R	R	R	Request version.
REQUEST = map	R				Request name.
REQUEST = GetMap		R	R	R	Request name.
LAYERS = layer_list	R	R	R	R	Comma-separated list of one or more map layers. Optional (ver. 1.1, 1.1.1) if SLD parameter is present.
STYLES = style_list	R	R	R	R	Comma-separated list of one rendering style per requested layer. Optional if SLD parameter is present.
SRS = namespace:identifier	R	R	R		Spatial Reference System.
CRS = namespace:identifier				R	Spatial Reference System.
BBOX = minx,miny,maxx,maxy	R	R	R	R	Bounding box corners (lower left, upper right) in SRS units.
WIDTH = output_width	R	R	R	R	Width in pixels of map picture.
HEIGHT = output_height	R	R	R	R	Height in pixels of map picture.
FORMAT = output_format	R	R	R	R	Output format of map.
TRANSPARENT = TRUE or FALSE	O	O	O	O	Background transparency of map (default = FALSE).
BGCOLOR = color_value	O	O	O	O	Hexadecimal red-green-blue color value for the background color (default = 0xFFFFFF).
EXCEPTIONS = exception_format	O	O	O	O	The format in which exceptions are to be reported by the WMS (default = XML).
TIME = time		O	O	O	Time value of layer desired.
ELEVATION = elevation		O	O	O	Elevation of layer desired.
Other sample dimensions		O	O	O	Values of other dimensions as appropriate.
Vendor specific parameters	O	O	O	O	Vendor specific parameters

WMS GetFeatureInfo Request

Request Parameter	1.0	1.1	1.1.1	1.3.0	Description
WMTVER = 1.0.0	R				Request version.
VERSION = version		R	R	R	Request version.
REQUEST = feature_info	R				Request name.
REQUEST = GetFeatureInfo		R	R	R	Request name.
<map_request_copy>	R	R	R	R	Partial copy of the Map request parameters that generated the map for which information is desired
QUERY_LAYERS = layer_list	R	R	R	R	Comma-separated list of one or more layers to be queried.
INFO_FORMAT = output_format	O	O	O	R	Return format of feature information (MIME type).
FEATURE_COUNT = number	O	O	O	O	Number of features about which to return information (default = 1).
X = pixel_column	R	R	R		X coordinate in pixels of feature (measured from upper left corner = 0)
I = pixel_column				R	i coordinate in pixels of feature in Map CS
Y = pixel_row	R	R	R		Y coordinate in pixels of feature (measured from upper left corner = 0)
J = pixel_row				R	j coordinate in pixels of feature in Map CS
EXCEPTIONS = exception_format		O	O	O	The format in which exceptions are to be reported by the WMS (default = XML).
Vendor-specific parameters	O	O	O		Optional experimental parameters.

WMS GetCapabilities

```
1 http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
2 da70dc92f2cc/services/ogc/wms?
3     SERVICE=wms&
```

```
    REQUEST=GetCapabilities&
    VERSION=1.1.1
```

Live Link

```
<?xml version='1.0' encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://schemas.opengis.net/wms/1.1.1/
WMS_MS_Capabilities.dtd"
 [
 <!ELEMENT VendorSpecificCapabilities EMPTY>
 ]>  <!-- end of DOCTYPE declaration -->

<WMT_MS_Capabilities version="1.1.1">

<!-- MapServer version 6.0.3 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML SUPPORTS=PROJ
SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=ICONV SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT
SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER
INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE -->

<Service>
  <Name>OGC:WMS</Name>
  <Title>tl_2010_35_state10</Title>
  <Abstract>WMS Service for RGIS dataset State Boundary - 2010
  (6ca5428a-a78c-4c82-8120-da70dc92f2cc)</Abstract>
        <KeywordList>
          <Keyword>RGIS</Keyword>
          <Keyword> New Mexico</Keyword>
        </KeywordList>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
  da70dc92f2cc/services/ogc/wms"/>
  <ContactInformation>
    <ContactPersonPrimary>
      <ContactPerson>GStore Support</ContactPerson>
      <ContactOrganization>Earth Data Analysis Center</ContactOrganization>
    </ContactPersonPrimary>
      <ContactPosition>technical support</ContactPosition>
    <ContactAddress>
        <AddressType>Mailing address</AddressType>
        <Address>Earth Data Analysis Center, MSC01 1110,
        1 University of New Mexico</Address>
        <City>Albuquerque</City>
        <StateOrProvince>NM</StateOrProvince>
        <PostCode>87131</PostCode>
        <Country>US</Country>
    </ContactAddress>
      <ContactVoiceTelephone>(505) 277-3622</ContactVoiceTelephone>
      <ContactFacsimileTelephone>(505) 277-3614</ContactFacsimileTelephone>
  <ContactElectronicMailAddress>gstore@edac.unm.edu</ContactElectronicMailAddress>
  </ContactInformation>
  <Fees>None</Fees>
  <AccessConstraints>none</AccessConstraints>
</Service>

```

```
<Capability>
  <Request>
    <GetCapabilities>
      <Format>application/vnd.ogc.wms_xml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-
          8120-da70dc92f2cc/services/ogc/wms?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-
          8120-da70dc92f2cc/services/ogc/wms?"/></Post>
        </HTTP>
      </DCPType>
    </GetCapabilities>
    <GetMap>
      <Format>image/png</Format>
      <Format>image/gif</Format>
      <Format>image/jpeg</Format>
      <Format>image/png; mode=8bit</Format>
      <Format>image/tiff</Format>
      <Format>application/vnd.google-earth.kml+xml</Format>
      <Format>application/vnd.google-earth.kmz</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-
          8120-da70dc92f2cc/services/ogc/wms?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-
          8120-da70dc92f2cc/services/ogc/wms?"/></Post>
        </HTTP>
      </DCPType>
    </GetMap>
    <GetFeatureInfo>
      <Format>text/plain</Format>
      <Format>application/vnd.ogc.gml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-
          8120-da70dc92f2cc/services/ogc/wms?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-
          8120-da70dc92f2cc/services/ogc/wms?"/></Post>
        </HTTP>
      </DCPType>
    </GetFeatureInfo>
    <DescribeLayer>
      <Format>text/xml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
```

```
104           da70dc92f2cc/services/ogc/wms?"/></Get>
105           <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
106           xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
107           da70dc92f2cc/services/ogc/wms?"/></Post>
108         </HTTP>
109       </DCPType>
110     </DescribeLayer>
111     <GetLegendGraphic>
112       <Format>image/png</Format>
113       <Format>image/gif</Format>
114       <Format>image/jpeg</Format>
115       <Format>image/png; mode=8bit</Format>
116       <DCPType>
117         <HTTP>
118           <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
119           xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
120           da70dc92f2cc/services/ogc/wms?"/></Get>
121           <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
122           xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
123           da70dc92f2cc/services/ogc/wms?"/></Post>
124         </HTTP>
125       </DCPType>
126     </GetLegendGraphic>
127     <GetStyles>
128       <Format>text/xml</Format>
129       <DCPType>
130         <HTTP>
131           <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
132           xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
133           da70dc92f2cc/services/ogc/wms?"/></Get>
134           <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
135           xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-
136           da70dc92f2cc/services/ogc/wms?"/></Post>
137         </HTTP>
138       </DCPType>
139     </GetStyles>
140   </Request>
141   <Exception>
142     <Format>application/vnd.ogc.se_xml</Format>
143     <Format>application/vnd.ogc.se_inimage</Format>
144     <Format>application/vnd.ogc.se_blank</Format>
145   </Exception>
146   <VendorSpecificCapabilities />
147   <UserDefinedSymbolization SupportSLD="1" UserLayer="0" UserStyle="1" RemoteWFS="0"/>
148   <Layer>
149     <Name>tl_2010_35_state10</Name>
150     <Title>tl_2010_35_state10</Title>
151     <Abstract>WMS Service for RGIS dataset State Boundary - 2010
152     (6ca5428a-a78c-4c82-8120-da70dc92f2cc)</Abstract>
153     <KeywordList>
154      <Keyword>RGIS</Keyword>
155      <Keyword> New Mexico</Keyword>
156     </KeywordList>
157     <SRS>EPSG:4269</SRS>
```

```
    <SRS>EPSG:4326</SRS>
    <SRS>EPSG:4267</SRS>
    <SRS>EPSG:26913</SRS>
    <SRS>EPSG:26912</SRS>
    <SRS>EPSG:26914</SRS>
    <SRS>EPSG:26713</SRS>
    <SRS>EPSG:26712</SRS>
    <SRS>EPSG:26714</SRS>
    <SRS>EPSG:3857</SRS>
    <LatLonBoundingBox minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
    <BoundingBox SRS="EPSG:4326"
                minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
    <Layer queryable="1" opaque="0" cascaded="0">
        <Name>tl_2010_35_state10</Name>
        <Title>tl_2010_35_state10</Title>
        <Abstract>State Boundary - 2010</Abstract>
        <KeywordList>
          <Keyword></Keyword>
        </KeywordList>
        <SRS>epsg:4326</SRS>
        <LatLonBoundingBox minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
        <BoundingBox SRS="epsg:4326"
                    minx="-109.05" miny="31.3322" maxx="-103.002" maxy="37.0003" />
        <MetadataURL type="FGDC-STD-001-1998">
          <Format>text/xml</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
          xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/meta
        </MetadataURL>
        <Style>
          <Name>default</Name>
          <Title>default</Title>
          <LegendURL width="72" height="22">
             <Format>image/png</Format>
             <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
             xlink:href="http://gstore.unm.edu/apps/rgis/datasets/6ca5428a-a78c-4c82-8120-da70dc92f2cc/s
          </LegendURL>
        </Style>
    </Layer>
  </Layer>
</Capability>
</WMT_MS_Capabilities>
```

WMS GetMap

```
http://gstore.unm.edu/apps/rgis/datasets/
6ca5428a-a78c-4c82-8120-da70dc92f2cc/
services/ogc/wms?
    VERSION=1.1.1&
    SERVICE=WMS&
    REQUEST=GetMap&
    BBOX=-109,31,-102.9,37.1&
    LAYERS=tl_2010_35_state10&
    WIDTH=200&
```

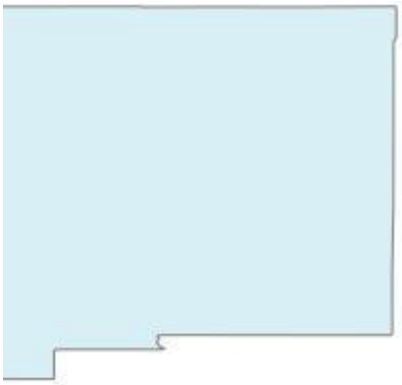


Figure 6.3: Sample WMS response #1

```
    HEIGHT=200&
    SRS=EPSG:4326&
    FORMAT=image/jpeg&
    STYLES=
```

link

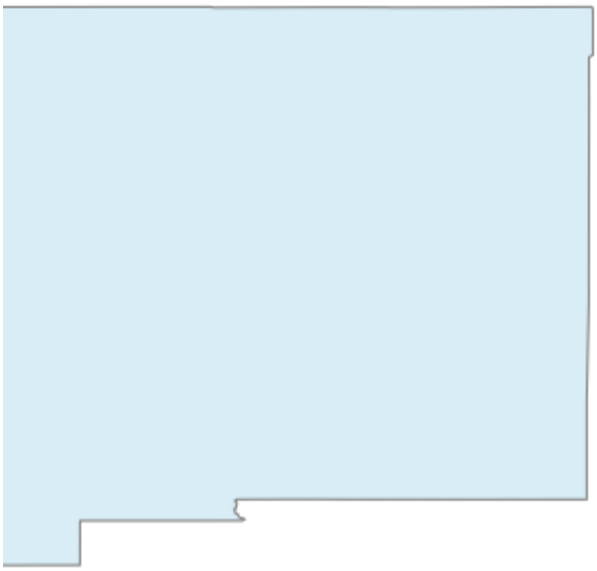


Figure 6.4: Sample WMS response #2

```
http://gstore.unm.edu/apps/rgis/datasets/
6ca5428a-a78c-4c82-8120-da70dc92f2cc/
services/ogc/wms?
    VERSION=1.1.1&
    SERVICE=WMS&
    REQUEST=GetMap&
    BBOX=-109,31,-102.9,37.1&
    LAYERS=tl_2010_35_state10&
    WIDTH=300&
```

```
    HEIGHT=300&
    SRS=EPSG:4326&
    TRANSPARENT=TRUE&
    FORMAT=image/png&
    STYLES=
```

link

Integraton of WMS and KML

- The KML GroundOverlay element may be used to integrate a network accessible map image into a client
- A WMS service may be used to as the source of a KML GroundOverlay element
- KML includes parameterizations that allow for dynamic generation of WMS requests using client bounding box information
- Time-enabled WMS may be accessed through use of manually configured time parameters in WMS URLs and TimeStamp or TimeSpan KML elements

Sample WMS-KML Integration

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2"
    xmlns:kml="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom">
    <GroundOverlay>
        <name>RGIS Counties WMS</name>
        <Icon>
            <href>http://gstore.unm.edu/apps/rgis/datasets/107046/services/ogc/wms?
            VERSION=1.1.1&amp;amp;SERVICE=WMS&amp;amp;REQUEST=GetMap&amp;amp;BBOX=-109,31,-102.9,37.1
            &amp;amp;LAYERS=tl_2010_35_state10&amp;amp;WIDTH=800&amp;amp;HEIGHT=800&amp;amp;SRS=EPSG:4326
            &amp;amp;FORMAT=image/png&amp;amp;STYLES=</href>
            <viewRefreshMode>onStop</viewRefreshMode>
        </Icon>
        <LatLonBox>
            <north>37.32753828398865</north>
            <south>30.86418272137246</south>
            <east>-101.3630220689848</east>
            <west>-110.6891149310152</west>
        </LatLonBox>
    </GroundOverlay>
</kml>
```

Sample KML File

Chapter 7

Week 7 - Module 4a - Interoperability Standards. WFS & WCS

Overview

- OGC Web Feature Services (WFS)
  - Capabilities and purpose
  - Overview of the collection of WFS commands
  - Sample WFS requests
- OGC Web Coverage Services (WCS)
  - Capabilities and purpose
  - Overview of the collection of WCS commands
  - Sample WCS requests

OGC Web Feature Service (WFS)

Background

The documents related to the OGC WFS standard are available from: `http://www.opengeospatial.org/standards/wfs` and all operation parameter tables presented here are based upon the OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

From the Version 2.0.0 WFS Scope Section:

> This International Standard specifies the behaviour of a service that provides transactions on and access to geographic features in a manner independent of the underlying data store. It specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions.
>
> Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.
>
> Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.

Locking operations allow exclusive access to features for the purpose of modifying or deleting features.

Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.

Stored query operations allow clients to create, drop, list and described parameterized query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

WFS Requests/Operations

These request types are submitted as part of the required `REQUEST` key in a KVP HTTP GET request.

`GetCapabilities` service metadata (XML) that documents the types of features supported by the service and the operations supported by each feature type
`DescribeFeatureType` metadata (XML) that describes the structure of supported feature types
`GetPropertyValue` a request for the value(s) of a specified property for a specified *featuretype*
`GetFeature (GetFeatureWithLock)` a request for actual features (XML, or other formats) from the service. The request may include both spatial and non-spatial query constraints
`LockFeature` Feature locking operation
`Transaction` a request to a WFS that may create, update, or delete features
`CreateStoredQuery` a request to create a named WFS query that is stored on the server for future reuse
`DropStoredQuery` a request to remove a named WFS query that has previously been stored on the server
`ListStoredQueries` a request to retrieve a list of named WFS queries that have been stored on the server
`DescribeStoredQueries` a request for more detailed information about specific named WFS queries that are stored on the server

WFS Conformance Levels

WFS 2.0.0 Requests and their corresponding WFS Compliance Levels

Operation (`REQUEST=`)	V 1.1.0	V 2.0.0	Simple	Basic	Transactional	Locking
`GetCapabilities`	✓	✓	✓	✓	✓	✓
`DescribeFeatureType`	✓	✓	✓	✓	✓	✓
`ListStoredQueries`		✓	✓	✓	✓	✓
`DescribeStoredQueries`		✓	✓	✓	✓	✓
`GetFeature`	✓	✓	✓	✓	✓	✓
`StoredQuery`		✓	✓	✓	✓	✓
`GetPropertyValue`		✓		✓	✓	✓
`Transaction`	✓	✓			✓	✓
`GetFeatureWithLock`	✓	✓				✓
`LockFeature`	✓	✓				✓
`GetGMLObject`	✓					

Request Composition

Requests submitted to a WFS may be submitted either via

HTTP GET a request that includes all request parameters within the URL submitted to the service. Request parameters are included in the URL as "key=value" pairs (KVPs)

HTTP POST a request where the URL consists of only the Host and path, with all other request parameters included in the body of the POST document submitted to the service. The request parameters supplied to the server are encoded as XML within the POST document.

SOAP a request submitted as an encapsulated message within a SOAP transaction.

Servers implementing WFS may support either the HTTP GET, POST, or SOAP request model

Conceptually *FeatureType* = *Layer*

KVP for Base WFS Requests

Base request parameters for all HTTP GET KVP requests (Figure 7.1)

Table 4 — KVP-encoding of the base request type

URLComponent	Operation	O/M[a]	Description
SERVICE	All operations.	M	See 7.6.2.4.
VERSION[b] (All operations)	All operations except GetCapabilities.	M	See 7.6.2.5.

[a] O = Optional, M = Mandatory

[b] VERSION is mandatory for all operations except the GetCapabilities operation.

Figure 7.1: Table 4 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

`VERSION` is required for all operations *except* the `GetCapabilities` request

Sample GetCapabilities Requests

Sample request to USGS Framework Layer (Governmental Units) WFS Service advertised by the USGS *TNM Access API page* service list - Live Link

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    request=GetCapabilities&
    service=WFS
```

Sample request to NM RGIS (NM 2010 Census Block Groups) - Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/715663ba-c1c3-414c-84a7-c671526f8316/services/ogc/wfs?
    SERVICE=wfs&
    REQUEST=GetCapabilities&
    VERSION=1.0.0
```

KVP for DescribeFeatureType Request

`DescribeFeatureType` HTTP GET KVP request (Figure 7.2)

Table 15 — DescribeFeatureType KVP encoding

URL Component	O/M[a]	Description
Common Keywords (REQUEST=DescribeFeatureType)		See Table 7. (Only keywords for all operations or the DescribeFeatureType operation.)
TYPENAME	O	A comma separated list of feature types to describe. If no value is specified, the complete application schema offered by the server shall be described.
OUTPUTFORMAT	O	Shall support the value "application/gml+xml; version=3.2" indicating that a GML (see ISO19136:2007) application schema shall be generated. A server may support other values to which this International Standard does not assign any meaning.
[a] O = Optional, M = Mandatory		

Figure 7.2: Table 15 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

Sample DescribeFeatureType Requests

USGS Framework Layer (Governmental Units) WFS Service linked from the USGS *TNM Access API page* service list - Live Link

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    version=1.1.0&
    request=DescribeFeatureType&
    service=WFS&
    typeName=WFS_govunits:State_or_Territory_High-res
```

Sample request to NM RGIS (NM 2010 Census Block Groups) - Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/715663ba-c1c3-414c-84a7-c671526f8316/services/ogc/wfs?
    VERSION=1.0.0&
    SERVICE=wfs&
    REQUEST=DescribeFeatureType&
    TYPENAME=tl_2010_35_bg10
```

KVP for GetFeature Request

`GetFeature` HTTP GET KVP request (Figure 7.3)

Table 17 — Keywords for GetFeature KVP-encoding

URL Component	Description
Common Keywords (REQUEST=GetFeature)	See Table 7 for additional parameters that may be used in a KVP-encoded GetFeature request.
Standard Presentation Parameters	See Table 5.
Standard Resolve Parameters	See Table 6.
Adhoc Query Keywords (Mutually exclusive with Stored Query Keywords)	See Table 8.
Stored Query Keywords (Mutually exclusive with Adhoc Query Keywords)	See Table 10.

Figure 7.3: Table 17 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

Table 5 — KVP-encoding of standard presentation parameters

URLComponent	Operation	O/M[a]	Default	Description
STARTINDEX	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.4.
COUNT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	1	See 7.6.3.5.
OUTPUTFORMAT	DescribeFeatureType, GetPropertyValue, GetFeature, GetFeatureWithLock	O	application/gml+xml; version=3.2	See 7.6.3.7.
RESULTTYPE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	results	See 7.6.3.6.

[a] O = Optional, M = Mandatory

Figure 7.4: Table 5 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

Table 6 — KVP encoding of standard resolve parameters

URLComponent	Operation	O/M[a]	Default	Description
RESOLVE	GetPropertyValue, GetFeature, GetFeatureWithLock	O	None	See 7.6.4.4.
RESOLVEDEPTH	GetPropertyValue, GetFeature, GetFeatureWithLock	O	*	See 7.6.4.5. RESOLVE parameter shall have a value other than "none".
RESOLVETIMEOUT	GetPropertyValue, GetFeature, GetFeatureWithLock	O	Server Specific (see ResolveTimeoutDefault, Table 14)	See 7.6.4.6. RESOLVE parameter shall have a value other than "none".
[a] O = Optional, M = Mandatory				

Figure 7.5: Table 6 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

KVP for GetFeature Request - Presentation Parameters (Figure {7.4})

KVP for GetFeature Request - Resolve Parameters (Figure {7.5})

KVP for GetFeature Request - Ad-hoc Query Parameters (Figure {7.6})

KVP for GetFeature Request - Stored Query Parameters (Figure {7.6})

Sample GetFeature Requests

USGS Framework Layer (Governmental Units) WFS Service linked from the USGS *TNM Access API page* service list - Live Link

Note: `TYPENAME` for `VERSION=1.1.0` instead of `TYPENAMES` for `VERSION=2.0.0`

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    VERSION=1.1.0&
    REQUEST=GetFeature&
    SERVICE=WFS&
    TYPENAME=WFS_govunits:State_or_Territory_High-res
```

Alternative request (Live Link) that includes an `OUTPUTFORMAT` parameter

```
http://services.nationalmap.gov/arcgis/services/WFS/govunits/MapServer/WFSServer?
    VERSION=1.1.0&
    REQUEST=GetFeature&
    SERVICE=WFS&
    TYPENAME=WFS_govunits:State_or_Territory_High-res&
    OUTPUTFORMAT=text/xml;%20subType=gml/3.1.1/profiles/gmlsf/1.0.0/0
```

Table 8 — Keywords for Ad hoc query KVP-encoding

URL Component	O/M[a]	Description
TYPENAMES	M[b]	See 7.9.2.4.1.
ALIASES	O	See 7.9.2.4.3.
SRSNAME	O	See 7.9.2.4.4.
Projection clause	O	See Table 9.
FILTER	O	See ISO 19143:2010, 6.3.3.
FILTER_LANGUAGE	O	See ISO 19143:2010, 6.3.3.
RESOURCEID	O	See ISO 19143:2010, 6.3.3.
BBOX	O	See OGC 06-121r3.
SORTBY	O	See ISO 19143:2010, Clause 8 The SORTBY parameter is used to specify a list of property names whose values should be used to order (upon presentation) the set of feature instances that satisfy the query. The value of the SORTBY parameter shall have the form *"PropertyName [ASC|DESC][,PropertyName [AASC|DESC],]"* where the letters ASC are used to indicate an ascending sort and the letters DESC are used to indicate a descending sort. If neither ASC nor DESC are specified, the default sort order shall be ascending. An example value might be: *"SORTBY=Field1 DESC,Field2 DESC,Field3"*. In this case the results are sorted by Field 1 descending, Field2 descending and Field3 ascending

a. O = Optional , M = Mandatory

b. The TYPENAMES parameter is mandatory in all cases except when the RESOURCEID parameter is specified (see 7.9.2.4.1).

Figure 7.6: Table 8 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

Table 10 — Keywords for Stored query KVP-encoding

URL Component	O/M[a]	Description
STOREDQUERY_ID	M	The identifier of the stored query to invoke.
storedquery_parameter=value	O	Each parameter of a stored query shall be encoded in KVP as a keyword-value pair. Stored query parameters shall not have names that conflict with any WFS parameter name.

a O = Optional, M = Mandatory

Figure 7.7: Table 10 from OpenGIS Web Feature Service 2.0 Interface Standard - Panagiotis (Peter) A. Vretanos, editor - 2010-11-02

OGC Web Coverage Services

Background

The documents related to the OGC WCS standard are available from: [http://www.opengeospatial.org/standards/wcs][wcs] with the sample parameters in the following slides based upon the *OGC Web Coverage Service 2.0 Interface Standard - KVP Protocol Binding Extension* - Peter Baumann, editor - 2010-10-27

From the OGC WCS 2.0 *Introduction*

> The OGC Web Coverage Service (WCS) supports electronic retrieval of geospatial data as "coverages" – that is, digital geospatial information representing space/time-varying phenomena.
>
> This document specifies the WCS core; every implementation of a WCS shall adhere to this standard. This standard thus defines only basic requirements. Extensions to the core will define extensions to meet additional requirements, such as the response encoding. Indeed, additional extensions are required in order to completely specify a WCS for implementation.
>
> A WCS provides access to coverage data in forms that are useful for client-side rendering, as input into scientific models, and for other clients. The WCS may be compared to the OGC Web Feature Service (WFS) and the Web Map Service (WMS). As WMS and WFS service instances, a WCS allows clients to choose portions of a server's information holdings based on spatial constraints and other query criteria.

WCS Requests/Operations

`GetCapabilities` service metadata (XML) that documents the service, including brief information about the data coverages available from the service

`DescribeCoverage` a request for more detailed metadata (XML) for one or more coverages listed in the output of the GetCapabilities request

`GetCoverage` a request for an actual data product representing a specified coverage. The specific data formats available for delivery will vary from service to service.

Request Composition

Requests submitted to a WCS may be submitted either via the following protocols, as defined in the three extensions developed thus far for the *core* WCS standard.

HTTP GET a request that includes all request parameters within the URL submitted to the service. Request parameters are included in the URL as "name=value" pairs. Extension Link

HTTP POST a request where the URL consists of only the Host and path, with all other request parameters included in the body of the POST document submitted to the service. The request parameters supplied to the server are encoded as XML within the POST document. Extension Link

XML/SOAP a request-response model between the client that conforms with the W3C SOAP web services protocol Extension Link

KVP for Base WCS Requests

Name	Mandatory/Optional	Definition	Data Type
service	M	Identifier of the OGC service	`String`, fixed to "WCS"
request	M	Request type name	`String`, set to operation name
version	M (except for GetCapabilities)	Request protocol version	`String`

Sample WCS `GetCapabilities` requests

NOAA Global Forecast System THREDDS catalog. Live Link

```
http://nomads.ncdc.noaa.gov/thredds/wcs/gfs-004/201602/20160228/
    gfs_4_20160228_0000_384.grb2?
    service=WCS&
    version=1.0.0&
    request=GetCapabilities
```

New Mexico Resource Geographic Information System PRISM Precipitation Normals WCS Service. Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/2ce10b57-3925-4971-b876-b6fc66d3cca2/services/ogc/wcs?
    SERVICE=wcs&
    REQUEST=GetCapabilities&
    VERSION=1.1.2
```

KVP for DescribeCoverage Request

`DescribeCoverage` HTTP GET KVP request (Figure 7.8)

Table 1 — DescribeCoverage request URL encoding

Name	Definition	Data type	Multiplicity
`service`	Identifier of the OGC service	`String`, fixed to "WCS"	One (mandatory)
`version`	Request protocol version	`String`	One (mandatory)
`request`	Request type name	`String`, fixed to "DescribeCoverage"	One (mandatory)
`coverageId`	List of coverage identifiers to be described	Comma-separated `NCName` list	One (mandatory)

Figure 7.8: Table 1 from *OGC Web Coverage Service 2.0 Interface Standard - KVP Protocol Binding Extension* - Peter Baumann, editor - 2010-10-27

Sample DescribeCoverage Request

NOAA Global Forecast System THREDDS catalog. Live Link

```
http://nomads.ncdc.noaa.gov/thredds/wcs/gfs-004/201602/20160228/
gfs_4_20160228_0000_384.grb2?
    service=WCS&
    version=1.0.0&
    request=DescribeCoverage&
    COVERAGE=Categorical_Rain
```

New Mexico Resource Geographic Information System PRISM Precipitation Normals WCS Service. Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/2ce10b57-3925-4971-b876-b6fc66d3cca2/services/ogc/wcs?
    SERVICE=wcs&
    REQUEST=DescribeCoverage&
    VERSION=1.1.2&
    COVERAGE=us_ppt_1971_2000_11
```

KVP for GetCoverage Request

`GetCoverage` HTTP GET KVP request (Figure 7.8)

Table 2 — *GetCoverage* request KVP encoding

Name	Definition	Data type	Multiplicity
`service`	Identifier of the OGC service	`String`, fixed to "WCS"	one (mandatory)
`version`	Request protocol version	`String`	one (mandatory)
`request`	Request type name	`String`, fixed to "GetCoverage"	one (mandatory)
`coverageId`	Identifier of coverage to be inspected	`NCName`	one (mandatory)
`subset`	boundaries of coverage subset	`SubsetSpec` as defined in Requirement 7	zero or more (optional)

Figure 7.9: Table 2 from *OGC Web Coverage Service 2.0 Interface Standard - KVP Protocol Binding Extension* - Peter Baumann, editor - 2010-10-27

Subset Definition for GetCoverage Request

Subset definition for the `GetCoverage` HTTP GET KVP request (Figure 7.10)

Requirement 7 /req/get-kvp/getCoverage-request-subsetspec:
Each `SubsetSpec` **shall** adhere to this EBNF syntax:

```
SubsetSpec:      dimension [ , crs ] ( intervalOrPoint )
dimension:       NCName
crs              anyURI
intervalOrPoint: interval | point
interval:        low , high
low:             point | *
high:            point | *
point:           number| "token" // " = ASCII 0x42
```

Figure 7.10: Requirement 7 from *OGC Web Coverage Service 2.0 Interface Standard - KVP Protocol Binding Extension* - Peter Baumann, editor - 2010-10-27

Example from the 2.0 specification:

```
http://www.myserver.org:port/path?
    service=WCS
```

```
    &version=2.0
    &request=GetCoverage
    &coverageId=C0002
    &subset=lon,http://www.opengis.net/def/crs/EPSG/0/4326(-71,47)
    &subset=lat,http://www.opengis.net/def/crs/EPSG/0/4326(-66,51)
  &subset=t,http://www.opengis.net/def/trs/ISO- 8601/0/Gregorian+UTC("2009-11-06T23:20:52Z")
```

Sample GetCoverage Request

New Mexico Resource Geographic Information System PRISM Precipitation Normals WCS Service. Live Link

```
http://gstore.unm.edu/apps/rgis/datasets/2ce10b57-3925-4971-b876-b6fc66d3cca2/services/ogc/wcs?
    SERVICE=wcs&
    REQUEST=GetCoverage&
    VERSION=1.1.2&
    COVERAGE=us_ppt_1971_2000_11&
    CRS=urn:ogc:def:crs:EPSG::4326&
    BBOX=24.0625,-125.02083333333,49.93749998965,-66.47916669008&
    FORMAT=image/tiff&
    WIDTH=2048&
    HEIGHT=905
```

Chapter 8

Week 10 - Module 2b - OpenLayers 3 Javascript Framework

Overview

- Capabilities
- OpenLayers = Javascript (by example)

OpenLayers Capabilities

- Support for Multiple basemaps: *BingMaps*, *MapQuest*, *OpenStreetMap*, *Stamen*
- Model for interaction with multiple map server platforms: *ArcGIS* (REST), *MapServer*, *GeoServer*
- Support for key OGC standards: *WMS*, *WMTS*, *WFS*, *GML*, *KML*
- Multiple control types: *Attribution*, *Zoom*, *Overview*, *Scale*, *FullScreen*, *Graticule*
- Custom styled features with associated attributes: *Curve*, *LinearRing*, *LineString*, *MultiLineString*, *MultiPoint*, *MultiPolygon*, *Point*, *Polygon*
- Support for many formats for data read and write: *ATOM*, *GML* (1, 2, 3), *GeoJSON*, *GPX*, *KML*, *WKT*, any many others
- Open Source, enabling modification and integration into other systems (e.g. GeoExt)

Distinguishing Characteristics Between OpenLayers and Google Maps

- Greater emphasis on client-side processing - Client access and rendering of data files that Google's servers otherwise take care of (pros & cons to this approach)
- Integrated support for OGC services and their products
- Support for different projections (adds complexity)
- API more rich in options ==> more complexity

Resources

OpenLayers Home Page

Application Programming Interface (API) Reference

Examples

Demonstrations and Examples

- Basic Mapper (with MapQuest base map (source))

OpenLayers_01.html

```
<html>

<head>
    <link rel="stylesheet" href="css/OpenLayers_01.css" type="text/css">
    <link rel="stylesheet" href="http://openlayers.org/en/v3.14.2/css/ol.css" type="text/css">
    <!-- you can use this line if you want to use the hosted version instead of the local copy -->
    <!-- <script src="http://openlayers.org/en/v3.14.2/build/ol.js" type="text/javascript"></script> --
    <script src="js/v3.14.2/build/ol.js" type="text/javascript"></script>
</head>

<body>
    <h1>This is a very simple OpenLayers 3 sample map page</h1>

    <div id='map'><!-- This is where the map will be displayed --></div>

    <!-- import the external Javascript file with the map configuration code -->
    <script src="js/OpenLayers_01.js" type="text/javascript"></script>
</body>

</html>
```

OpenLayers_01.js

```
// OpenLayers_01.js

var myMap = new ol.Map({
    target: 'map',
    layers: [
        new ol.layer.Tile({
        source: new ol.source.MapQuest({layer: 'sat'})
        })
    ],
    view: new ol.View({
        center: ol.proj.fromLonLat([-106.624083,35.08427]),
        zoom: 18
        })
    });
```

OpenLayers_01.css

```
/* OpenLayers_01.css */

body {
    width:100%;
    hwight:100%
}
```

```
7
8  #map {
9      width:600px;
10     height:400px
11 }
```

Demonstration and Examples - Online Resources

- Mapper (source) with a variety of base maps (MapQuest, Stamen, OSM) and basic layer selection
- Basic Mapper with Controls: No Controls (source), Customized Controls (source)

Next Week - Custom Features and WMS Layers

Chapter 9

Week 11 - Module 2b - OpenLayers Javascript Framework

Overview

- More detailed Map Object Options
- More detailed Layer Object Options
- Additional Map Layer Types - With Examples

Map Object Options

- Map Object Options API Reference
- View Object Options API Reference
- Layer Object Options
  - `ol.layer.Tile` API Reference
  - `ol.layer.Image` API Reference
  - `ol.layer.Vector` API Reference
  - `ol.layer.VectorTile` API Reference

A variety of strategies for constructing a new `OpenLayers.Map` object

```
// create a map with minimum required elements and default
// options in an element with the id "map1"
var map = new ol.Map({
    target:'map1',
    // a map without layers can be defined and in that case a map with no layers
    // will be rendered
    layers: [
        new ol.layer.Tile({
            source: new ol.source.MapQuest({layer: 'osm'})
        })
    ], // end layers
    view: new ol.View({
        center: [0, 0],
        zoom: 1
```

```
    }), //end view
});

// create a map with options specified in a separate 'options' variable and
// included by reference in the code to create the new map object
var options = {
    // required options
    target:'map2',
    layers: ...,
    view: ...,

    // optional options - only include those that you need
    controls: ...,
    pixelRatio: ...,
    interactions: ...,
    keyboardEventTarget: ...,
    loadTilesWhileAnimating: ...,
    loadTilesWhileInteracting: ...,
    logo: ...,
    overlays: ...,
    renderer: ...
};
var map = new ol.Map(options);

// map with non-default options - same as above but with a single argument
var map = new ol.Map({
    // required options
    target:'map2',
    layers: ...,
    view: ...,

    // optional options - only include those that you need
    controls: ...,
    pixelRatio: ...,
    interactions: ...,
    keyboardEventTarget: ...,
    loadTilesWhileAnimating: ...,
    loadTilesWhileInteracting: ...,
    logo: ...,
    overlays: ...,
    renderer: ...
});

// the following commands can be executed to add, set or remove the layers in a map
// after a map object has been created

map.addLayer(layer)
map.removeLayer(layer)
map.setLayerGroup(layerGroup)

// the view of a layer can be created or modified after the map object has been
// created by using the following command

map.setView()
```

```
69
70 // the target DOM object for the map object can be set or changed using
71 // the following command
72
73 map.setTarget
```

Layer Object Options

Layer Types and a subset of sources for each type

- `ol.layer.Image` - a single map image is rendered for this layer type
  - `ol.source.ImageMapGuide` - API source is a MapGuide server hosting data of interest.
  - `ol.source.ImageStatic` - API source renders a specified static image file within a specified extent within the map.
  - `ol.source.ImageWMS` - API source retrieves a single map image from the specified OGC Web Map Service (WMS).
- `ol.layer.Tile` - map images in a tiled grid are rendered for this layer type
  - `ol.source.TileArcGISRest` - API source is an ArcGIS REST map or image service
  - `ol.source.TileWMS` - API source is an OGC Web Map Service (WMS)
  - `ol.source.WMTS` - API source is an OGC Web Map Tile Service (WMTS)
- `ol.layer.VectorTile` - map content is delivered vector data that has been divided into a tile grid and cannot be edited
  - `ol.source.VectorTile` - API source delivers vector data tiles for rendering in the client (example
- `ol.layer.Vector` - map content is delivered as vector data that is rendered by the client and may be edited within the client
  - `ol.source.Vector` - API the source for vector feature(s) that constitute a vector layer. The individual features are `ol.Feature` objects that consist of at least one geometry, or a collection of geometries and any additional attributes that are associated with each feature.

Common Pattern of Layer Object Creation (varies some depending upon the specific layer type)

```
1 var layer = new ol.layer.***({
2     source: new ol.source.***({
3         ...
4     }),
5     other options ...
6 })
```

Additional Map and Layer Object Functions & Events

Both Map and Layer Objects have a number of associated functions as well

- Retrieving object properties programmatically with `Get` functions.
- Modifying existing object properties with `Set` functions
- Map destruction, and reconfiguration
- Linkage of object events with Javascript functions

WMS Layer Configuration

Some key issues to be aware of when using the two WMS supporting layers (`ol.layer.Tile`, and `ol.layer.Image`) and their associated WMS sources (`ol.source.TileWMS` and `ol.source.ImageWMS` respectively) include:

- The *projection* of the map object must be supported by the included WMS service (review the WMS GetCapabilities response to see what projections are supported by the service). If you don't specify a `projection` parameter as part of the map object's `view` property a default *Web Mercator* (EPSG:3857) projection is used for the map. Information about how to define and set map projections in OpenLayers is found here
- The *layers* parameter as part of the `params` option must be provided as part of the server-related property list (the layer names may also be found in the GetCapabilities response)
- Other WMS parameters (again as part of the `params` option) may be provided as well to "adjust" the request automatically generated by OpenLayers
- Use of a tiled WMS may produce unwanted repetition of labels included in the WMS. If that is the case you can use a single-image `ol.layer.Image` layer type to allow the WMS server to handle the distribution of layers across the entire map image instead of including them in each individual map image.

Sample WMS Layer Object Creation

```
var basemap_single = new ol.layer.Image({
    source: new ol.source.ImageWMS({
        attributions: new ol.Attribution({
            html: 'Blue Marble Next Generation:' +
            'R. Stockli, E. Vermote, N. Saleous, R. Simmon and D. Herring (2005). The Blue Marble Next (
        }),
        params: {'LAYERS':'global:BMNG_west'},
        url: 'http://mapper.karlbenedict.com:8080/geoserver/global/wms?',
        serverType: 'geoserver'
    })
})

var states_single = new ol.layer.Image({
    source: new ol.source.ImageWMS({
        attributions: new ol.Attribution({
            html: 'State Boundary Restructured - USGS, National Atlas Release 5-14-12'
        }),
        params: {'LAYERS':'global:statep010'},
        url: 'http://mapper.karlbenedict.com:8080/geoserver/global/wms?',
        serverType: 'geoserver'
    })
})

var singleMap = new ol.Map({
    target: 'map_image',
    layers: [basemap_single,states_single],
    view: new ol.View({
        center: ol.proj.fromLonLat([-98.58,39.83]), // the approximate geographic center of the contine
        zoom: 3,
        projection: 'EPSG:3857'
        })
```

```
    });


```

Example: HTML, Javascript

Vector Layer Configuration

Vector layers support

- External Data in a Variety of supported formats for both *reading* and *writing* (just a sample): GML, GPX, GeoJSON, JSON, KML, WFS, WKT, Open Streetmap
- Directly encoded geometries: Circle, Geometry, GeometryCollection, LinearRing, LineString, MultiLineString, MultiPoint, MultiPolygon, Point, Polygon, SimpleGeometry
- User created features, including support for interactive editing of features
- Styling of Vector features

Sample Point Feature Object creation

```
var classroomCoord = [-106.624073,35.084280]
var officeCoord = [-106.624899,35.084506]

var classroomPoint = new ol.geom.Point(classroomCoord);
var officePoint = new ol.geom.Point(officeCoord);
```

Sample KML Layer Object creation with style

```
var blocks_kml = new ol.layer.Vector({
    source: new ol.source.Vector({
        url: 'https://s3.amazonaws.com/kkb-web/data/tl_2010_35001_tabblock10.kml',
        projection: projection,
        format: new ol.format.KML()
    })
})

var county_style = new ol.style.Style({
    fill: new ol.style.Fill({
      color: county_color
    }),
    stroke: new ol.style.Stroke({
      color: county_color,
      width: 1
    }),
});

var counties_kml_styled = new ol.layer.Vector({
    source: new ol.source.Vector({
        url: 'https://s3.amazonaws.com/kkb-web/data/2007fe_35_county00.kml',
        projection: projection,
        format: new ol.format.KML({'extractStyles':false}),
        style: county_style
```

```
25      })
26  })
27
```

Example: HTML, Javascript

Chapter 10

Module 4b - Interoperability Standards - Desktop GIS Integration

Overview

- Common Model for Client Configuration for Connections to Remote OGC Services
- Specific Client Examples

Quantum GIS (QGIS)

- WMS
- WFS
- WCS

ArcGIS

- WMS
- WFS
- WCS

Common Model

Based upon the results of a GetCapabilities request against a remote service. GetCapabilities request information provided as either:

- The base URL to which the OGC service parameters would be added
- A complete GetCapabilities request against the service

Full GetCapabilities Request

NASA Earth Observations (NEO) Imagery WMS

`http://neowms.sci.gsfc.nasa.gov/wms/wms?version=1.3.0&service=WMS&request=GetCapabilities`

USGS Service Endpoints - http://services.nationalmap.gov/arcgis/services/USGSTopoLarge/MapServer/WMSServer?request=GetCapabilities&service=WMS

Base URL for GetCapabilities

NASA Earth Observations (NEO) Imagery WMS

http://neowms.sci.gsfc.nasa.gov/wms/wms?

USGS Service Endpoints - http://services.nationalmap.gov/arcgis/services/USGSTopoLarge/MapServer/WMSServer?

Quantum GIS (QGIS)

QGIS uses the Base URL approach for adding WMS, WFS or WCS layers to a project.

The General Process:

1. Add service, or select existing service
2. Connect to the service to retrieve the information from the GetCapabilities response for the service
3. Select layer(s)
4. Modify settings for layer(s)
5. Add layer(s)

QGIS OGC Documentation

QGIS - Adding Services and Layers - start

You need to know the GetCapabilities request for the service you want to add, for example one of the USGS WMS services

```
http://services.nationalmap.gov/arcgis/services/USGSTopoLarge/MapServer/WMSServer?request=GetCapabiliti
```

determine the base URL -

```
http://services.nationalmap.gov/arcgis/services/USGSTopoLarge/MapServer/WMSServer
```

in this case

If in doubt, check the information in the metadata

Select the layer type you would like to from “Layer” menu, or click the button in the interface to add a specific layer type.

QGIS - Adding Services and Layers - adding a service

Add a service to the list of services in the menu (if necessary - QGIS retains information about previously added services) by selecting the “New” option under the service list in the “Add Layer(s) from a Server” dialog

QGIS - Adding Services and Layers - adding connection information

Add the name, base URL and any additional information about the service to the connection dialog box

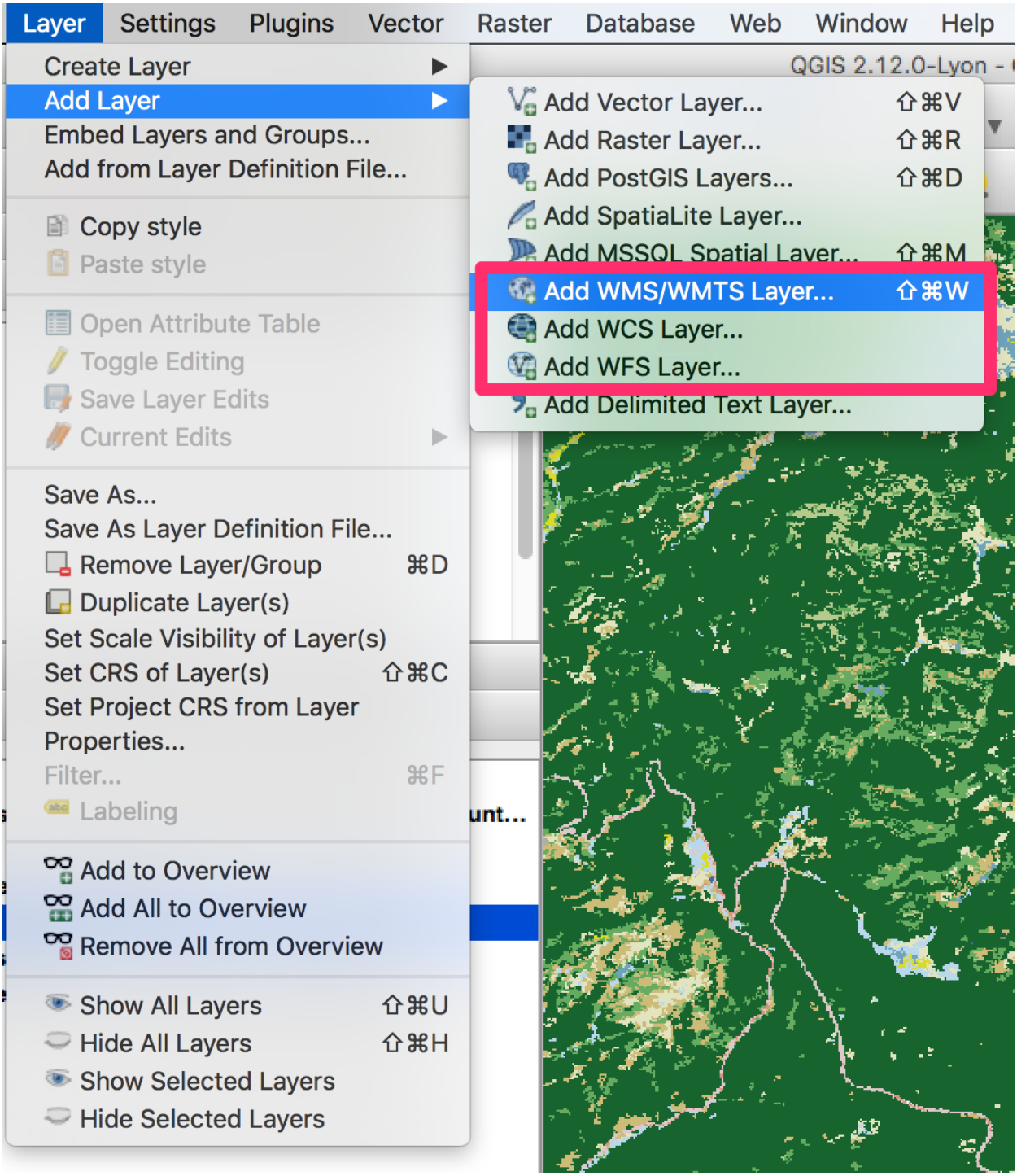


Figure 10.1: OGC Options in the *Add Layer* menu options

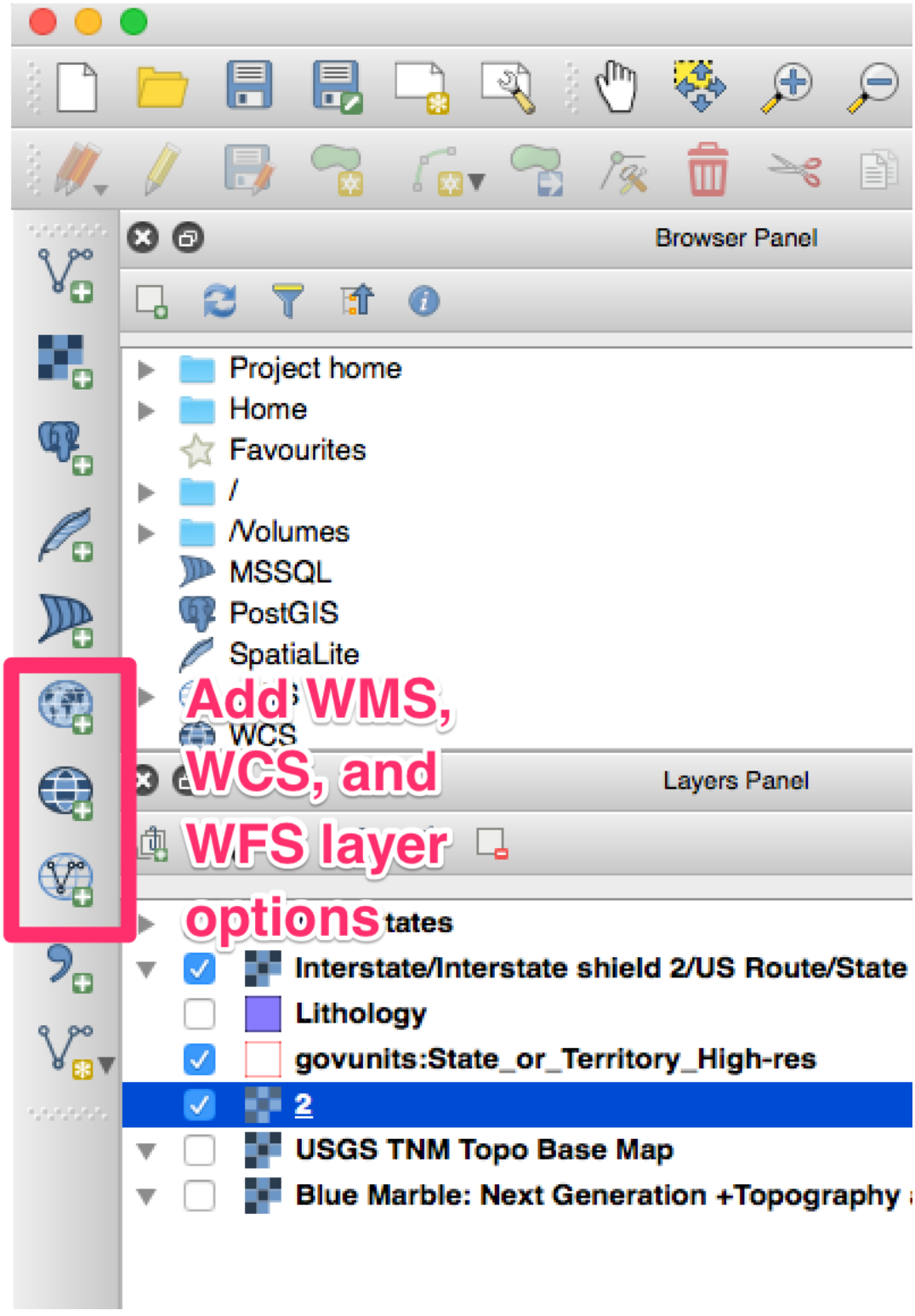


Figure 10.2: OGC Options in the left-side icon list

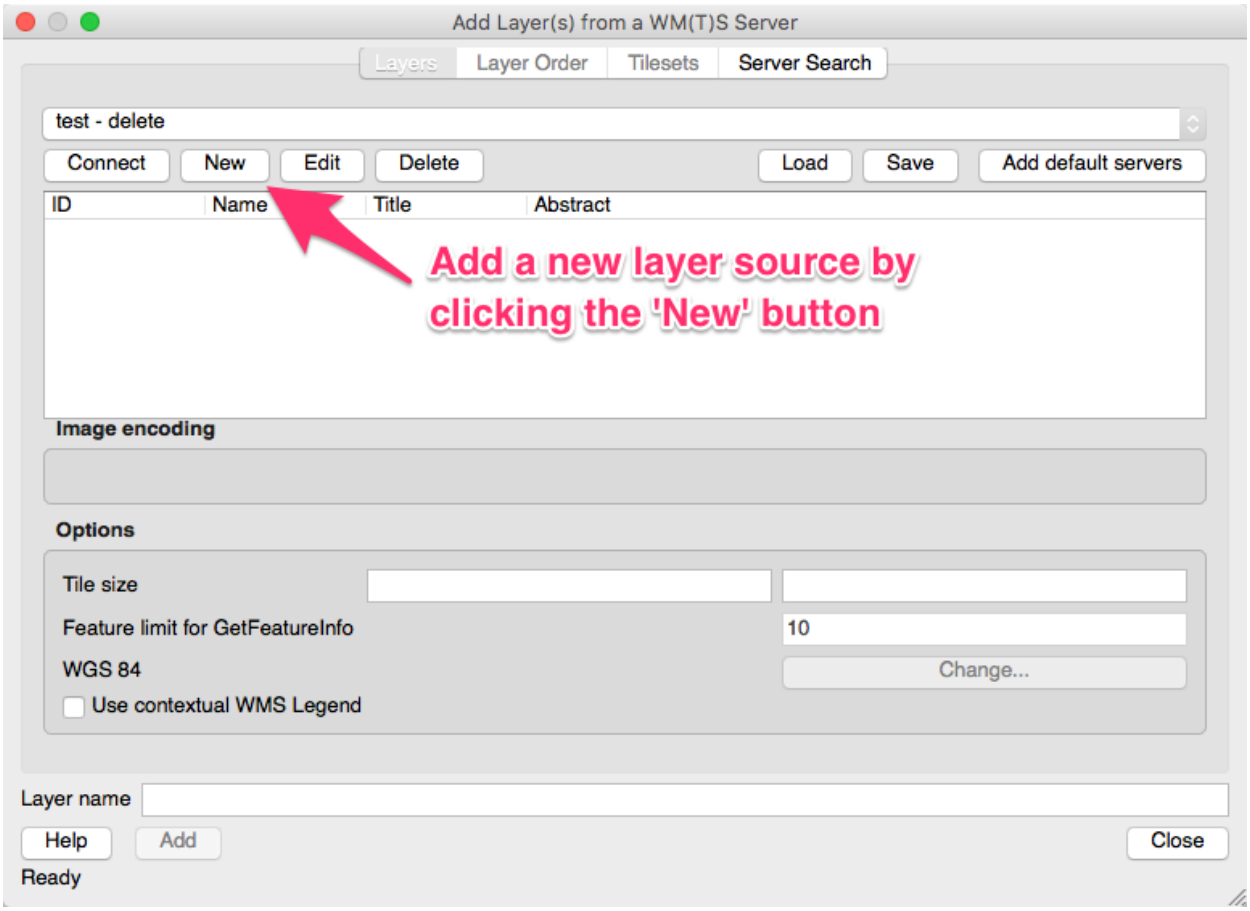


Figure 10.3: "Add Layer(s) from a Server" dialog

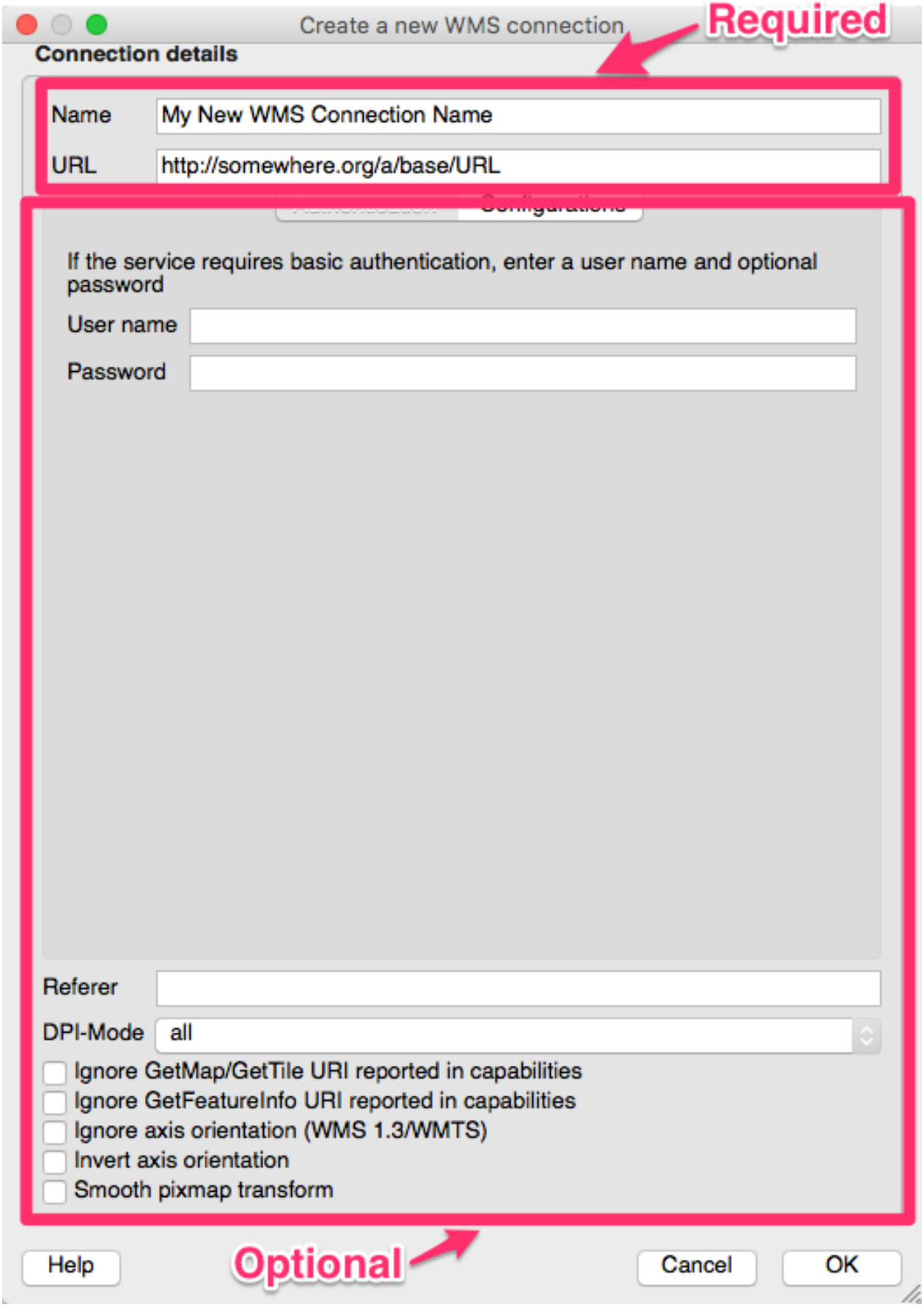


Figure 10.4: Connection dialog box

QGIS - Adding Services and Layers - connecting to and adding layers from the service

After adding the service, you can select it from the service list in the "Add Layer(s) from a Server" dialog box, connect to the service to retrieve the GetCapabilities response from the service, select the layers and other options advertised by the service through its response, and add them to your map.

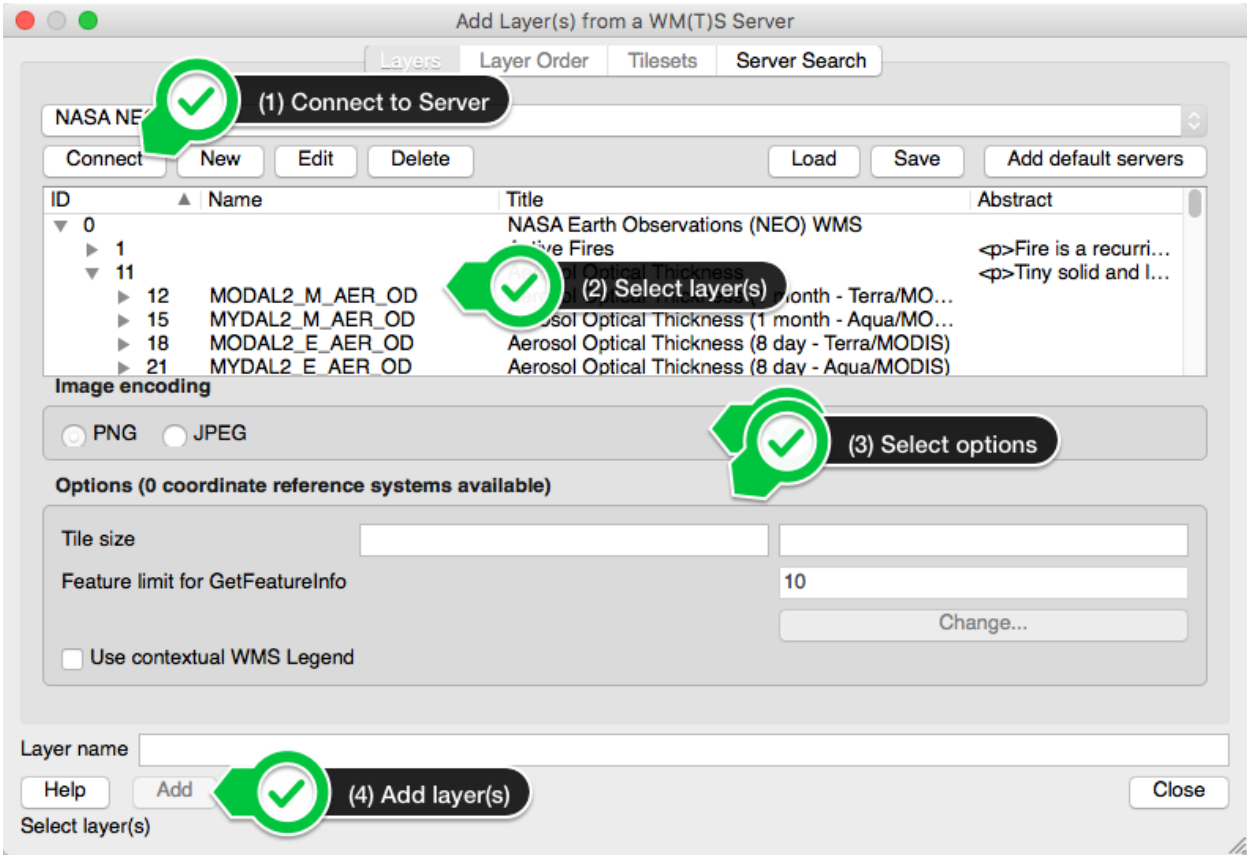


Figure 10.5: "Add Layer(s) from a Server" dialog with service connection and layer selection process

QGIS - Adding Services and Layers - the final added layer

After adding the layer, it appears as an available layer in the table of contents for your map.

QGIS Demonstration with WMS, WFS and WCS Services

WMS, WFS and WCS in QGIS

Example

ArcGIS

Based upon the results of a GetCapabilities request against a remote service. GetCapabilities request information provided as either:

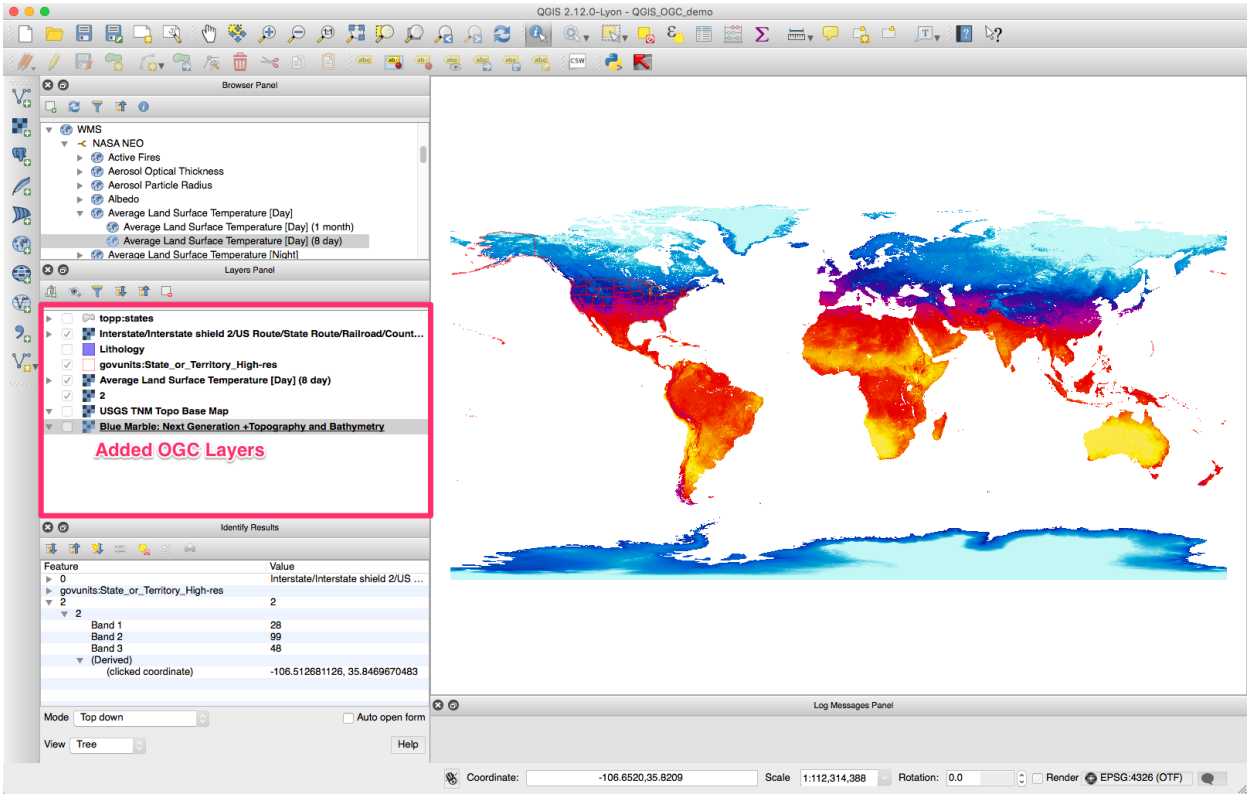


Figure 10.6: QGIS map with added WMS layer in the TOC

- The base URL to which the OGC service parameters would be added
- A complete GetCapabilities request against the service

This model applies to ArcGIS just as it did for Quantum GIS - the base URL is provided to the various ArcGIS components that support the addition of OGC services to the client interface.

ArcGIS OGC Support Documentation

ArcGIS WMS and WCS Configuration

The addition of OGC WMS and WCS layers to ArcMap is through the same process of

- Select the "add data" button
- WMS/WCS services are added through the "GIS Servers" option in the "Add Data" dialog
- If you have not previously added the service from which you want to add layers, you select "Add WMS Server" or "Add WCS Server" from the list of options in the "Add Data" dialog
  - You then provide the BASE GetCapabilities URL to the "ADD WMS/WCS Server" dialog that appears
  - Click "OK", and the new WMS/WCS service is added to the list of services that is available when you choose to add a WMS service.
- You then select the layer(s) from the service that you want to add to your map and click the "add" button in the dialog.

ArcGIS WMS and WCS Configuration Resources

- Adding WMS Services to ArcMap 10.2

ESRI Documentation

- Adding WCS Services to ArcMap 10.2

ESRI Documentation

ArcGIS WFS Configuration

- WFS support in ArcGIS 10.0 and beyond requires that the "Data Interoperability Extension" be installed (though it doesn't have to be enabled)
- Connections to WFS services are defined through ArcCatalog's "Interoperabilty Connections" "Add Interoperability Connection" option
- After defining the connection in ArcCatalog (including the specification of the interoperabilty connection type, desired feature types, and maximum number of features to return), its feature types are available through that Interoperability Connection that may be added to ArcMap and other ArcGIS components
- Once the connection is created, WFS data may be added through the "Add Data" dialog in ArcMap

ArcGIS WFS Configuration Resources

- Steps for connecting to an OGC WFS from within ArcCatalog 10.2

ESRI Documentation

- Steps for adding a WFS service to ArcMap 10.2

ESRI Documentation

Conclusions

- A GetCapabilities request is the key for configuring most OGC client applications to access remote services
- The specific way in which the GetCapabilities request is given to the client varies from client to client
- Clients can auto/*mis*-configure themselves based upon the GetCapabilities XML response - when troubleshooting problems with an advertised service, try the manual request approach for the GetCapabilities, data and maps that you have learned about to determine if the service is functioning as advertised.
