Exercise 1: Static Locations, Dynamic Content  
In this exercise, we will learn how to configure the following:

* The Basic Map Frame
* Marker Overlays
* Info Windows & HTML content
* Stream Gauge Overlays

## Step One: Getting Started

In this first step, we’ll open up a basic .html page, change some of the parameters, and see how it looks. To prep for this exercise, do the following:

1. Navigate to the training data, and under the “Ex1” folder, and double-click the “ex1final.html” to open the file in a browser. Click on the markers. This is what we’ll be working towards in this exercise. Close the browser.
2. Inside the Ex1 folder, copy the “ex1\_step1.html” file and paste it in the same folder. Rename it “ex1\_<yourname>.html”. This will be the file you will be editing through the rest of the exercise.
3. Right-click “ex1\_<yourname>.html” and choose “Edit” or “Open With” and choose Notepad or Notepad++ to edit the file. This should open the page in a text editor.

### Explore the code

Explore the code inside the file, looking for the items below.

#### DOCTYPE and html, head, title, and script tags

The DOCTYPE establishes this file as HTML5, and most modern browsers should read this file just fine. Following that are the opening tags for <html> and <head>, and a <meta> tag. After that is the opening and closing <title> tag, of which you can change the contents if you wish. The first <script> tag loads up the Google Maps API at version 3, and tells the API that this is not a location-aware page. The next <script> tag is where we’re going to do all our work.

#### Style and Body sections

Scroll down below the second </script> tag. The <style> section just gives some declarations that tell the browser that the html and body of this document should take up 100% of the window. After this section, the end </head> tag follows, then the <body> section, which establishes “divs” for the map and a small section called “details” where we’ll be placing some useful text.

### Make Edits to the Code

Right now, this file has containers, but no actual map. Find the initialize function and type the following in between the { } brackets:

var latlng = new google.maps.LatLng(45.00,-95.75);

var myOptions = {

zoom: 10,

center: latlng,

mapTypeId: google.maps.MapTypeId.ROADMAP

};

map = new google.maps.Map(document.getElementById("map\_canvas"),myOptions);

Save your edits, then preview the file in a browser. Keep the browser window open, and then modify the parts in gray above to suit your preferences. Refresh the browser each time you make a change to view the difference:

1. Change the lat/long to a point of your interest. An easy way to obtain this is to use Google Maps at maps.google.com or look up a city in Wikipedia.
2. Change the zoom level. Increase it to zoom in closer, or decrease it to see more area. (The most practical levels for Minnesota will likely be between 6 and 14.)
3. Change the MapType to one of the following to see the effect: SATELLITE, HYBRID, or TERRAIN.

Optional: change the width and height percentages in the styles of the map\_canvas and details divs to see what you prefer:

<div id="map\_canvas" style="width**:**100%**;** height**:**80%">This is where the map will go</div>

<div id="details" style="width**:**100%"></div>

## Step Two: Add the Stream Gage Marker

In this step, we’ll add some in order to add markers to the map.

### Find Stream Gage Information

Before adding stream gage overlays, you need to find a stream gage you’re interested in. Visit this site to find a gage near your area of interest: <http://www.dnr.state.mn.us/waters/csg/index.html>

Click on the dots on the map to see details of the gage you’re interested in. Write down the name of the gage and its unique site ID (an 8-digit number like 28054001 or 82035001). Then use the CSV file “csg\_sites.csv” in the training folder to obtain the site’s lat/long value (e.g. 44.200166, -94.194226 – you can find the site in the CSV file using the station ID). You will use this information in the next section.

### Add Variables for the Stream Gage

Above the initialize function, add three variables that will hold the stream gage information, like this:

**var** csgName = "Site Name"; //type in your site name here

**var** csgID = 22007001; // type in your ID here

**var** csgLatLong = **new** google.maps.LatLng(45.0206, -95.8689); //replace the lat/longs with your stream gage

Each “var” statement above establishes a variable we will use in the next section to add a marker. If you want to try a different stream gage later, all you have to do is return to this section of code and change the variable definitions.

### Add the marker definition

Inside the initialize function, after establishing the map, add the following to create a marker for the stream gage, like this:

**var** marker = **new** google.maps.Marker({

position:csgLatLong,

map: map

});

This establishes and adds the marker for our stream gage. In the next step, we’ll use the name and ID as well as the marker. Preview your file in a browser to make sure you can see your marker appearing on the map. You may have to pan and zoom it lies outside the extent of your initial map – feel free to go back to the initialize function and make adjustments to the initial LatLng and zoom level to adjust for this.

## Step Three: Create Info Window

In this step, we’ll set up some base variables that are common for all the stream gage info windows and create an info window function that can be called after the code that creates our marker.

### Create Base Variables

Because we want dynamic information in our info window, we need some base variables that will establish today’s date and the date a week ago. This will let us show stream gage data from the previous week in our info window. While we’re at it, we’ll establish some image sizes for our graphs, and the base/tail URLs for our info window graphs. Above the initialize function and below the variables you modified for the stream gage, add the following base variables and definition statements:

**var** imgHeight = 300;

**var** imgWidth = 500;

**var** weekAgo = **new** Date();

**var** yesterday = **new** Date();

weekAgo.setDate(weekAgo.getDate()-7);

yesterday.setDate(yesterday.getDate()-1);

**var** gEndFullDate = yesterday.getFullYear() + "-" + (yesterday.getMonth()+1) + "-" + yesterday.getDate();

**var** gStartFullDate = weekAgo.getFullYear() + "-" + (weekAgo.getMonth()+1) + "-" + weekAgo.getDate();

**var** baseGraphURL = ["http://maps1.dnr.state.mn.us/cgi-bin/csg/hydrograph\_cgi.py?var1=232&show\_legend=1&show\_grid=1&site="];

**var** tailGraphURL = ["&start="+gStartFullDate+"&end="+gEndFullDate];

### Create InfoWindow Function

Below the initialize function, add a new function called createInfoWindow that receives the marker, ID, and csgName variable from the marker definition, like this:

**function** createInfoWindow(marker, ID, csgName) {

**var** myHtml = "<img src=\""+

baseGraphURL + ID + tailGraphURL +

"&height="+

imgHeight+

"&width="+

imgWidth+

"\" height=\""+

imgHeight+

"\" width=\""+

imgWidth+

"\">";

**var** infowindow = **new** google.maps.InfoWindow({

position: marker.getPosition(),

content: myHtml

});

}

This function establishes some basic HTML content of an image, with a source as a URL call to DNR’s stream gage server. The URL includes a base, the stream gage ID, a tail (begin and end dates), and height and width parameters. We also have to specify the height and width inside the <img> tag, so we simply match them using the imgWidth and imgHeight variables.

The function then creates an infowindow using the marker’s position as an anchor and adds the HTML content. To enable this function, add a after the marker definition, like this (insert the gray section):

**var** marker = **new** google.maps.Marker({

position:csgLatLong,

map: map

});

createInfoWindow(marker, csgID, csgName); //ONLY INCLUDE THIS LINE!

## Add a Listener

The last thing you need to do to enable your info window is to add a listener, which tells the map marker to “listen” to the “click” event, and open up the info window when clicked. Inside the createInfoWindow function, paste the following code:

google.maps.event.addListener(marker, 'click', **function**() {

infowindow.open(map,marker);

 });

Launch your page in a browser to make sure the info windows work – they’ll appear when you click on each marker, and should look something like the image at right.

## Challenge: Additional Details Functions

Launch the ex1final.html file in a browser and click on the marker. What is different between this and what you finished in Step 3? Examine the code of ex1final.html – how do we enable these additional functions?