Designing a GUI interface and Presenting Data in Google Maps API

Author: Paul Solomon

Date: April 5, 2013

Abstract

The Google Maps API provides a tool for easily visualizing map data. By using this tool, a graphical user interface (GUI) can be constructed that displays different types of data on a Google Map object. This data can come from an outside source and be implemented into Google Maps with the use of programming languages such as HTML, CSS, and Javascript. Other tools, such as jQuery and JSON can be used to update information on a Google Map in real time, making the map a real time indicator of data. Putting together all of these tools allows one to create a GUI on a website that is effective in displaying data in an easy to read way by the user.

Keywords

GUI, API, Google, Maps, Javascript, jQuery, Data
Introduction

What is a Graphical User Interface (GUI)?

A graphical user interface, or GUI, is a display that shows a user all of the relevant data and controls of a system. A GUI is often shown on a computer screen or some kind of display, such as a liquid crystal display (LCD). A user can input and manipulate data from the GUI in a simplistic way that does not require the user to know the entire workings of the system. GUIs should be easy to read and interact with without the need for excessive background knowledge. Examples of a GUI would be what you see on the screen when turning on a cell phone, or what you see when visiting a website.

When designing a GUI, it is important to keep in mind who the end user is that using your GUI. Differences in prior knowledge of the system or the subject material you are displaying will affect what kinds of data or controls you decide to put into your display. If your end user has an expansive knowledge of the data and the system, it would be important to display more details and advanced options. On the other hand, if an end user will not be as knowledgeable on the subject, fewer controls and objects on the screen would enhance the experience for that individual and not clutter the screen with non-useful information.

A popular graphical user interface that many people are familiar with using are internet websites. Websites offer an accessible way to view information that many people are comfortable using. When developing a GUI, websites offer a great platform that is incredibly flexible in what can be created. Websites are built on a number of programming languages, including HTML, CSS, Javascript, Java, and PHP. Using these tools, one can create an effective and visually pleasing GUI in little time.

What is an API?

When developing a website, it can be cumbersome to hand-code every detail of the website on your own. Fortunately, many software companies provide full libraries of functions, classes, and other data structures to make more complex systems with simple commands. These libraries are often referred to as application programming interfaces, or APIs. Software companies will often release APIs for free so that developers can program new features or content to existing pieces of software. APIs are released for specific programming languages, depending on the product.

Google Maps API

Background

Google Maps is a popular tool that Google has created to show map data all around the world. Google Maps can show all different kinds of data, including satellite images, road maps, locations of interest, and much more. Google has opened up its Google Maps software to developers by releasing an API so that web developers can create their own Google Maps websites that fit their needs. The API is written for Javascript programming of websites. The following information will show how to set up a Google Maps element on a website and how to add various types of data to it.
NOTE: This document assumes the reader has some preexisting knowledge of web programming. This includes some experience in object oriented programming, HTML, CSS, Javascript, jQuery, and JSON. For more information on these subjects, go to the following link, which provides excellent detailed tutorials and documentation about web programming:

http://www.w3schools.com/

**Getting Started**

To be able to work with Google Maps, you must first obtain a free API key provided by Google. The key is required to be able to run a Google Maps element on your website. Directions on how to obtain a key can be found here:

https://developers.google.com/maps/documentation/javascript/tutorial#api_key

Once a key has been obtained, you must link to the Javascript API in the head tag of your website HTML. Figure 1 shows a coding example of how to add the Google Maps API into HTML:

```html
<head>
<title>Map</title>
<link href="map-style.css" rel="stylesheet" type="text/css" />
<script src="jquery-1.9.1.min.js"></script>
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=YOURKEYHERE&libraries=visualization,geometry&sensor=false"></script>
<script src="map.js"></script>
</head>

*Figure 1: Adding the Javascript API to an HTML document*
```

Once this has been done, the Google Maps API will be accessible in your HTML document. The next step is to place it in the body of your HTML file. Create an empty `<div>` element with a CSS ID that you can access later. Figure 2 shows an example `<div>` element that would be placed in the body of your HTML document:

```html
<div id="map-canvas"></div>

*Figure 2: Empty `<div>` element where the Google Map will be placed*
```

By default, the size of the Google Map is 0 by 0 pixels. To change this, manually set the size property of the div element to the desired size. Figure 3 shows an example CSS element that sets the size of the Google Map:
The final step in placing the Google Map into a website is to initialize the Map object in Javascript with your desired options. To do this, first create a Javascript code segment that runs when the website first loads. A simple way to do this is to use the “ready” function in jQuery to call a start-up function.

```javascript
$(document).ready(function() {
  startup();
});
```

**Figure 4: jQuery "ready" function**

Then, in the `startup()` function, place the following lines of code:

```javascript
var mapOptions = {
  center: new google.maps.LatLng(42.726883,-84.48964),
  zoom: 19,
  mapTypeId: google.maps.MapTypeIds.SATELLITE
};

// Initialize map
map = new google.maps.Map(document.getElementById("map-canvas"), mapOptions);
```

**Figure 5: Javascript code to initialize the Google Map**

As you can see in Figure 5, there are a number of options that can be set when initializing a new Map object. The “center” option indicates where the Google Map will first locate itself on the globe, given a latitude and longitude. The “zoom” option is set to the initial zoom level of the map. This ranges from 0 to 22, with 0 meaning no zoom and a zoom level of 22 meaning zoomed into the map as far as it goes. The last option showed here, “mapTypeId”, is for setting what type of map to show. This can be set to a traditional map with roads, or set as “satellite” which will show satellite imagery.

There are a number of additional settings that can be set here, which can be found in the Google Maps documentation at this link: [https://developers.google.com/maps/documentation/javascript/reference#MapOptions](https://developers.google.com/maps/documentation/javascript/reference#MapOptions)

Also in Figure 5, the last line of code shown in that block finally initializes the map. In the `getElementById()` function, it is given an input: “map-canvas”. This is the ID of the `<div>` element that was created earlier in the HTML document. If done correctly, the map will look like what is shown in Figure 6:
Adding Markers to Google Maps

At this point, a Google Map object was created and placed into an HTML document with a number of desired parameters. From here, more information can be added, such as markers.

A useful object to add to a map is a marker to represent a position on a map. The Google Maps API makes placing markers quite simple. The following block of code shows the basic structure for creating a new marker on your map:

```javascript
var Latlng_temp = new google.maps.LatLng(42.7273065439394, -84.49025690807343);
var marker_temp = new google.maps.Marker(
    position: Latlng_temp,
    map: map,
    draggable: true,
    title: "Marker 1"
);
```

Figure 7: Creating a new marker

Similar to how options were set for the Map object, a Marker object contains a set of options that can be set when creating it. The first option, “position”, is set to what coordinates the marker should be placed at. The “map” option is required, and needs to be set to the Map object that was previously
created. In this example, the Map object was simply called `map`. The “draggable” parameter is set to either “true” or “false”, indicating if the new Marker object can be dragged around the screen by the user when clicked on. The last option shown in this example is “title”, which is used to give the marker a title so that when the user hovers over the marker, it will display its given title.

Also, like the Map object, there are multiple additional options that can be set to further alter the Marker object. The following link shows all of the available options given by the Google Maps API: https://developers.google.com/maps/documentation/javascript/reference#MarkerOptions

The same process can be used to make any number of Marker objects. Below is an example of the map with six different markers:

![Map with multiple Marker objects](image)

**Figure 8: Map with multiple Marker objects**

More Google Maps Objects

The Google Maps API has many more types of objects that can be created beyond just Markers. Two other types of objects that are commonly used are InfoWindows and Heatmaps. InfoWindows can be created to display information on a specific Marker when clicked. A bubble pops up and can display information of the users choosing. Heatmaps allow for a colored representation of data from Marker to Marker to be shown on the map. Both of these objects can be created in a similar fashion to the Marker object. Information regarding these objects can be found here: https://developers.google.com/maps/documentation/javascript/reference#InfoWindow https://developers.google.com/maps/documentation/javascript/reference#HeatmapLayer
Creating a GUI with Google Maps

Background

By this point, one can see how Google Maps can be a useful tool in plotting data on a map. However, the next question is how to utilize this with outside data and data that will be dynamically changing over time. There are a number of ways to accomplish this, but all require the same necessary steps. These steps are: importing data, parsing data, and updating the web page.

NOTE: For this section, it is assumed that a database has already been created and populated by an outside source. The following information will discuss only how to capture this data in your website code for use in Google Maps.

Importing Data

The first step in using outside data is to import it into your Javascript code as an object. One common form of holding data is the use of a JSON file. A JSON file can contain a database of information in an external file that can be imported into Javascript. The contents of an example JSON file are shown in Figure 10:
jQuery offers a useful function called $.getJSON() that retrieves data from an external JSON file. From here, the next step is to parse the JSON object into something that can be used.

Parsing Data

Another handy function in jQuery, called $.each(), can iterate through an object, such as JSON, and perform a function on each element in the object. Combining this with $.getJSON(), we can write some lines of code to import and iterate through a JSON file.

```javascript
$.getJSON("pagedata.json", function(result) {
  $.each(result.markers, function(i, markers) {
    // do something
  });
});
```

*Figure 11: Import and iterate through a JSON file*

Updating the Web Page

The next step is to update the data on the web page. One way to do this is to call a function at a set interval of time. The Javascript function to do this is setInterval().

```javascript
setInterval(function() {updateMeasurement()}, 3000);
```

*Figure 12: Calls the updateMeasurement() function every 3 seconds*

In the updateMeasurement() function, we will import the JSON data, parse the JSON data, then update elements in the map. An example shown in Figure 13 shows how to update the content inside of an infoWindow object, as well as updating a Heatmap object on a map.
This block of code works by populating a temporary table with the information in the JSON file. It then checks if this data is different from what is already shown on the web page. If it is the same, it will do nothing, but if the data is different, it will update the data in the infoWindow and Heatmap objects and refresh the page.

**Adding Additional Functionality**

From here, one can add any number of additional functionality to the Google Maps object to better create their GUI. Additional buttons and controls could be added for the user to manipulate the data or see the data in a different way, such as in a graph, as opposed to in only in a map. The Google Maps API offers a platform for building more and more complicated functionality that is certainly beyond what was covered in this document. The Google Maps API Javascript reference guide explains, in detail, all of the different functions and data that the API could potentially create: [https://developers.google.com/maps/documentation/javascript/reference](https://developers.google.com/maps/documentation/javascript/reference)