Maps-1
Google Maps

Overview

Mapping applications are very popular on desktop computers, and arguably one of the most popular and frequently used applications on mobile devices (after music, mail and web browsers).

MapBlast (now MSN Maps) was an early web mapping application, and Yahoo's MapQuest is still around, but Google Maps is now the 800 pound gorilla of online maps.

Our next two assignments are going to be using the Google Maps API.

i) You can find a nice history and overview of the capabilities of the Google Maps API at: http://en.wikipedia.org/wiki/Google_Maps

ii) The home page for the google maps API is here:
https://developers.google.com/maps/

iii) The home page for the google maps Web API is here:
https://developers.google.com/maps/web/

iv) The "Hello World" map tutorial is here:
https://developers.google.com/maps/documentation/javascript/tutorial

v) The Google Maps API, which lists properties, methods and events for classes like `Map`, `Marker` and `Polygon`, is here:
https://developers.google.com/maps/documentation/javascript/reference

vi) Lot's of sample code is here:
https://developers.google.com/maps/documentation/javascript/examples/

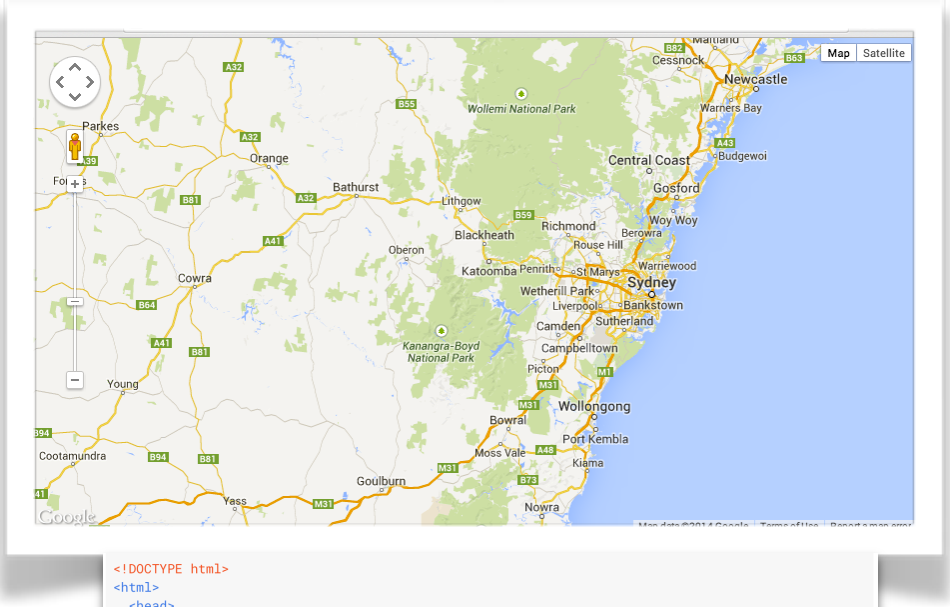
vii) The API signup page is here:
https://developers.google.com/maps/documentation/javascript/get-api-key

Part I. Google Maps - Markers and InfoWindows

1) Go ahead and sign up for the google maps API and get your API key:
https://developers.google.com/maps/documentation/javascript/get-api-key

2) Once you have your API key, go ahead and test it with the Hello, World example from the google site. Name it **map-simple.html**. Sydney, Australia should be visible on the map:

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




```html
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Map</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>
      html, body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
      #map {
        height: 100%;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
      var map;
      function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
          center: {lat: -34.397, lng: 150.644},
          zoom: 8
        });
      }
    </script>
    <script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
    async defer></script>
  </body>
</html>
```

The google map docs have a line-by-line explanation of what the code does, go ahead and read that over now.

A) Write below how you can specify the coordinate (latitude -80, longitude 22) as an object literal.

{lat: -80, lng: 22}

B) Which zoom level will show …

- the World fully zoomed out: 1
- Landmass/Continent: 5
- City: 10
- Streets: 15
- Buildings: 20

Part II. The semester is almost over and I need caffeine

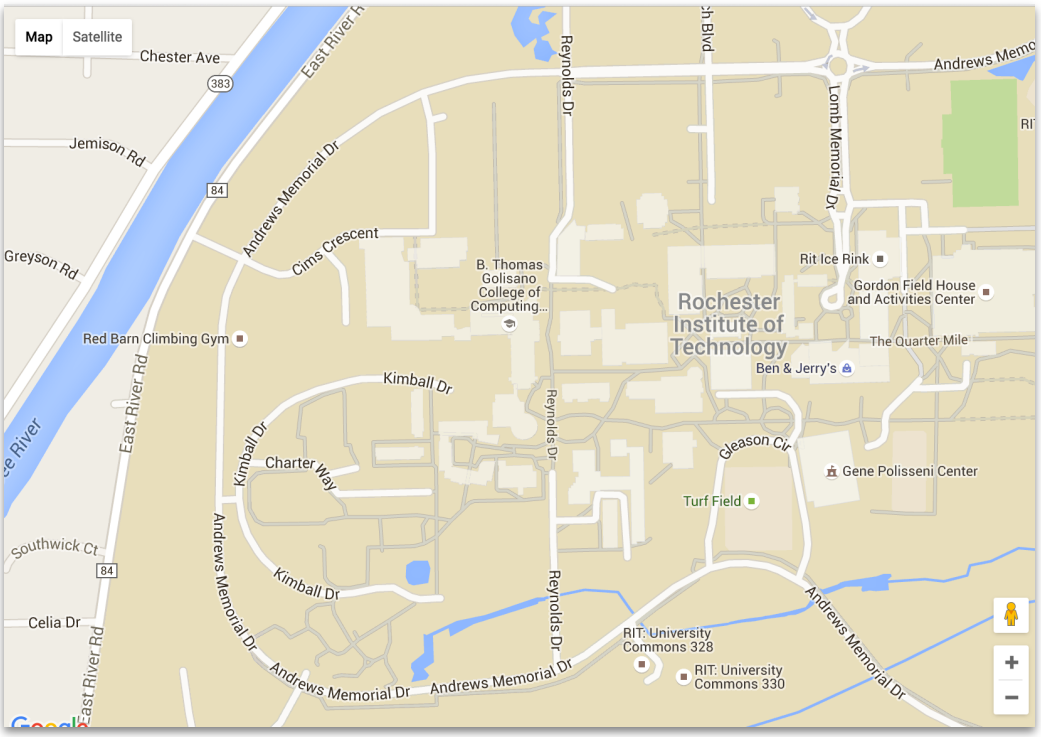
1) Duplicate **map-simple.html** and name the copy **rit-coffee-map.html**

Change our `initMap()` code to appear as follows:

```
<script>
  let map;
  function initMap() {
  let mapOptions = {
    center: {lat:43.083848,lng:-77.6799},
    zoom:16,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
    map = new google.maps.Map(document.querySelector('#map'), mapOptions);
  }
</script>
```

Note that we're changing the `center` property and `zoom` property of the map, and now specifying a `mapTypeID`.

Reload the page, it should now be centered and zoomed in on Golisano College.



2) We also could have set these properties directly though the `map` object. You can see the properties, methods, and events of the `Map` class here:

https://developers.google.com/maps/documentation/javascript/reference#Map

`map.setCenter()`, `map.setZoom()`, and `map.setMapTypeId()` are the methods you could use.

Let's add some simple controls to the map.

A) Here's the HTML that goes right after the `<div id="map"></div>` tag:

```
<p><button id="worldZoomButton">World Zoom (1)</button></p>
<p><button id="defaultZoomButton">Default Zoom (16)</button></p>
<p><button id="buildingZoomButton">Building Zoom (20)</button></p>
```

B) Here's the CSS:

```
button{
  position: absolute;
  color:red;
  font-weight:bold;
  height:30px;
  width: 130px;
  z-index: 100;
}

#worldZoomButton{
  top:70px;
  left:10px;
}

#defaultZoomButton{
  top:110px;
  left:10px;
}

#buildingZoomButton{
  top:150px;
  left:10px;
}
```

C) Here's the JS:

```
document.querySelector("#worldZoomButton").onclick = function(){
    map.setZoom(1);
};

document.querySelector("#defaultZoomButton").onclick = function(){
    map.setZoom(16);
};

 document.querySelector("#buildingZoomButton").onclick = function(){
    map.setZoom(20);
};
```

You should now have functional zoom buttons:

Here we have added standard HTML DOM elements as controls. Google also has their own set of controls that mesh nicely with the maps (scroll down to the controls section of this page for examples):

https://developers.google.com/maps/documentation/javascript/examples/

By the way:
Latitude is 0 at the equator, and gets larger as it moves north.
Longitude is 0 at Greenwich, UK (near London), and gets smaller moving west.

Greenwich's coordinates are 51.4800 N latitude, 0.0000 longitude. Longitude is calculated from this Greenwich Meridian.

4) Let's add a `Marker` to the map. A Marker is a kind of overlay on the map - you can think of it as a pin that identifies a location on the map. Here are the docs:

https://developers.google.com/maps/documentation/javascript/reference#Marker

The tutorial also covers Markers very well:

https://developers.google.com/maps/documentation/javascript/markers

A) To put a marker on our map at Crossroads, add the following code to `initMap()`

```
let position = {lat:43.082634, lng: -77.68004};
let marker = new google.maps.Marker({position: position, map: map});
marker.setTitle("Crossroads");
```

Reload the page, you should see the following:

B) Let's go ahead and load in 12 coffee locations on campus. You'll need to import the **coffee-data.js** file (use a `<script>` tag) This file contains a hard-coded array of objects named `coffeeShops`. There are 12 objects in the array, each of which represents the location and name of a coffee shop on campus. The objects look like this:

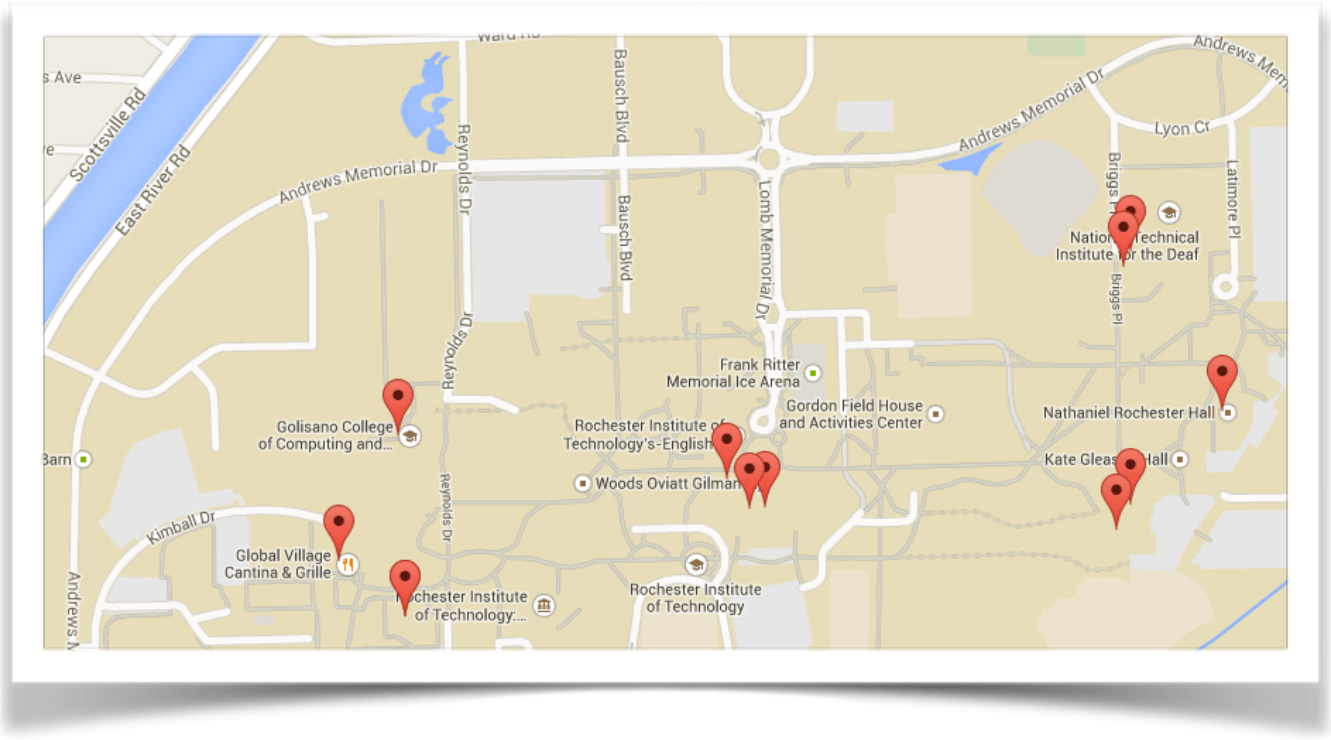
```
{
  latitude:43.084156,
  longitude:-77.67514,
  title:"Artesano Bakery & Cafe"
},

{
  latitude:43.083866,
  longitude:-77.66901,
  title:"Beanz"
},
…
```

C) Delete the previous Marker code, and write the following helper function for creating markers:

```
function addMarker(latitude, longitude, title) {
  let position = {lat:latitude,lng:longitude};
  let marker = new google.maps.Marker({position: position, map:map});
  marker.setTitle(title);
}
```

D) Now write code in `initMap()` that loops through `coffeeShops` and calls `addMarker()` using the values from the array. When you are done, you should see all 12 coffee locations.



F) An `InfoWindow` is bubble-like overlay that we can put text into. It's usually attached to a `Marker`. The InfoWindow docs are here: https://developers.google.com/maps/documentation/javascript/reference#InfoWindow

To get an InfoWindow to appear when we click on a marker, we'll need to start by declaring a *script scoped* variable named `infowindow`.

G) Then we'll create another helper function named `makeInfoWindow()`:

```
function makeInfoWindow(position,msg){
   // Close old InfoWindow if it exists
      if(infowindow) infowindow.close();

      // Make a new InfoWindow
      infowindow = new google.maps.InfoWindow({
               map: map,
               position: position,
               content: "<b>" + msg + "</b>"
      });
}
```

H) To add a click event handler to each marker as it is created, add the following to the end of `addMarker()`:

```
// Add a listener for the click event
google.maps.event.addListener(marker, 'click', function(e){
     makeInfoWindow(this.position,this.title);
});
```

Reload the page, clicking on a marker should cause a small window to appear that contains the name of the coffee shop.

I) To get 45-degree maps, add the following to `initMap()`:

```
map.mapTypeId = 'satellite';
map.setTilt(45);
```

Not all locations and zoom levels have 45-degree imagery - RIT's was just recently added.

J) Now go ahead and add an “Isometric View” button that zooms into zoom level 18. Zoom level 18 will show the 3D imagery.

Submission:

Upload your files to Banjo. Zip and post your code to the dropbox along with the links in the submission comments.

--- Optional Bonus on Next Page ---

Part III. Polygons - Optional

1) For 10 bonus points on this HW assignment, get Polygons drawing: https://developers.google.com/maps/documentation/javascript/reference#Polygon

   Polygons form a closed loop of coordinates and can be stroked and filled.

- load the **building-data.js** file
- write a helper function called `function drawPolygon(paths,title){…}`
- loop through the `buildings` array and draw the building Polygons (there are only 2 in the file, GOL and ORN)