Volume 13 (2013)

Interface: The Journal of Education, Community and Values

5-10-2013

Dotted Landscape: Berglund Center for Internet Studies Fellowship Review and Analysis Part 3

Michael Geraci
Pacific University

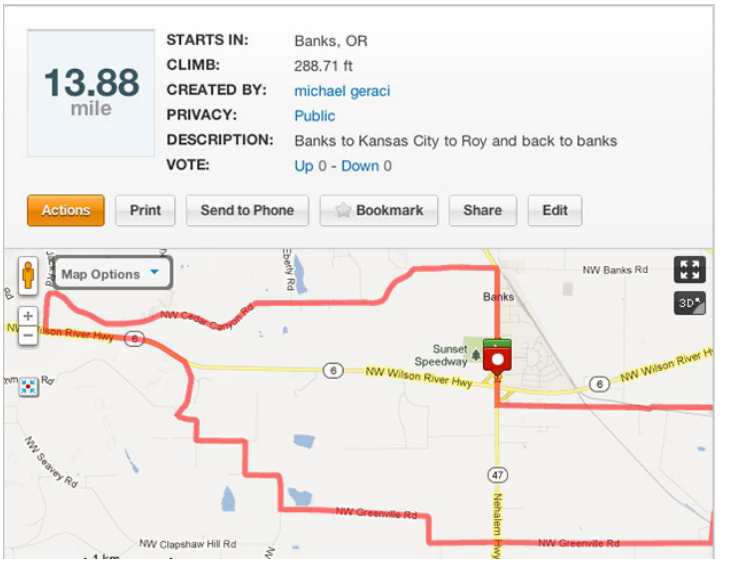
Dotted Landscape: Berglund Center for Internet Studies Fellowship Review and Analysis Part 3

Dotted Landscape: Berglund Center for Internet Studies Fellowship Review and Analysis Part 3

by Michael Geraci

The Technology Behind the Journey

There are a number of online tools and technologies that were combined in the making of the Dotted Landscape project. The planning of the rides themselves relied on a Web service known as Map My Ride. [1] Map My Ride allows cyclists to create and search for cycling routes all over the world. The site provides a wealth of information about cycling routes and produces printable guides. Relying on an innovative modification to Google's mapping system, riders can plot routes based on location, distance, terrain type, and difficulty. I used Map My Ride to create the ten routes that I would cover on my bike for the purposes of this project. You can go to Map My Ride and search on "Dotted Landscape" or "Michael Geraci" to find my



collection.

Figure 5. A screen shot of a cycling route on Map My Ride.

Map My Ride was just the beginning of the useful tools I came to use and appreciate in the making of this project. Another tool that I did not intend to use, but found invaluable, was an online conversion tool for the geographical data produced by my camera. When I started implementing my own mapping system to plot the location of the trash I learned, much to my chagrin, that Google's system relied on the WGS 84 format, which expresses geographical location in decimal format (e.g., 45.50983, -122.991 [1] for latitude and longitude, respectively). My camera, along with many other GPS devices, records location in the more conventional degrees, minutes, and seconds (e.g., 45,30.59N, 122,59.46W). This difference required that every geo-location that I had recorded had to be converted to Google's standard. Luckily, I was able to find numerous online sources for the conversion of this information. I employed a student research assistant to help me in running the conversions, which had to be done one at a time. We used two sites for this task: Computer Support Group, Inc.'s "GPS Latitude and Longitude Converter" [2] and iTouchMap.com's "Latitude and Longitude of a Point" service. [3]

With the data in the correct format, I set about learning how to embed Google maps on a Web page, set their initial view properties, and plot locations within them. The inclusion of Google maps on a site or page is a fairly common and well-documented practice in Web development, but the placement of multiple markers with embedded information that would be included in the markers' info windows necessitated my first in-depth study of JavaScript programming. I had used JavaScript in small amounts throughout my tenure as a Web developer, but this usage was for fairly common practices such as building rollover effects for buttons, and drop-down menus for site navigation. This new endeavor required that I learn to create code "objects" for each marker that contained all of the data used to create the map points for each piece of litter. Early versions of this meant writing code like:

```
var r4i1Marker = {title:"object title",id:49,ride:4,date: "September 27, 2011",location:{lat:45.509833,lng:122.991}, thumbnail:"imgs/thumb/r4i1.jpg",full:"imgs/full/r4i1.jpg", caption:"image caption",url:"item49.html"}
```

This format for encapsulating multiple properties for a single object enabled me (via JavaScript) to not only place markers in the interface, but aso for all aspects of their presentation to be self-contained in a single, portable code element that could be accessed by the browser. After successfully generating all the marker data for the first ride in the project, I felt that this system, while technically appropriate, was not the most efficient way to store and access the

information about the 133 trash items I would be featuring on the site. Further research was needed to find a more flexible and scalable system of structuring data. The research quickly yielded the use of AJAX, [4] Asynchronous JavaScript and XML, as the way to write a single function that would serve as the technological backbone of the entire project. But a move towards this efficiency meant more study.

AJAX is a common technology that drives many popular Web sites such as Facebook, Netflix, and just about all major Google services like maps, Gmail, and Google Drive. In a nutshell, it is a system of passing data back and forth between the user's browser and the Web server in real time, thereby eliminating the need for constant page loading. AJAX is not a language, but a mode of JavaScript programming that relies on an external data source written in XML, [5] or other standard data structures like JSON. [6]

I was no stranger to XML, as it is a common format for storing information that needs to be accessed by any application or device that can read it. My first use of XML was in the creation of RSS "feed" files that I created in 2006, when I began exploring the world of Podcasting [7] as an earlier bit of professional development as an educator. I set about stripping all of the object notation (as shown above) out of my JavaScript and converting it to standard XML. The object shown above soon became just a fraction of a growing XML library that I wrote for the project using the free text editing application, TextWrangler [8], which looked like this:

```
<item>
<title>Car Front Bumper</title>
<id>49</id>
<ride>4</ride>
<number>1</number>
<marker>r4i1Marker</marker>
<date>September 27, 2011</date>
<location>
<lat>45.509833</lat>
<lng>-122.991</lng>
<road>Hwy 219 South of Hillsboro</road>
</location>
<category>automotive</category>
<info>
<full>imgs/full/r4i1.jpg</full>
<thumb>imgs/thumb/r4i1.jpg</thumb>
<caption>
A silver bumper from the front of a small car
</caption>
</info>
<nar>
<url>item49.html</url>
```

```
<media>imgs/full/xl/r4i1.jpg</media>
</nar>
</item>
```

Storing the information for the site in this format allowed me to abstract my "back end" data from the site's code and access some or all of it by any number of JavaScript functions.

JavaScript is not a new language; it has been around since 1995. One of the main reasons why it has become so popular in recent years is because a coding eco-system has grown up around it that has made it easier to tap into the vast functionality without having to fully master all aspects of the language. The free jQuery code library [9] sits on top of the JavaScript behemoth and lets developers implement large-scale functionality on their websites in a few short lines of code. JQuery's own motto is "write less, do more" and, faced with the task of writing the engine for Dotted Landscape, this was appealing to me. Similar to my previous experiences with XML, I was no stranger to jQuery as it is the tool of choice for creating slick online photo galleries, animations, and other Web 2.0 experiences. However, in all of these cases, jQuery is easily plugged into a webpage and a few simple additions to HTML markup results in browser "magic" that do not necessarily require an in-depth understanding of what was going on behind the scenes. For the first time, I was faced with using jQuery to simplify the creation of custom site behavior that was not just plugged in. Luckily, jQuery is well documented online [10] and there are numerous sources for writing AJAX that drives functionality in Google maps. [11]

After a few weeks, the AJAX functionality was running and now a single jQuery function (all 132 lines and 3300 characters of it) was bringing my ten maps and their associated pieces of trash to life. Enthused by the successful implementation of this system, I decided to expand the site's core functionality to include the dynamic generation of the sidebar menus on the site that listed all the trash for the current ride (see figure 6). Extracting the titles from the XML file and building a listing of them was a simple matter. However, I wanted the user to be able to click an item in the menu and have the corresponding marker in the map interface open its info window. This small feat required learning how events (like mouse clicks) in one part of the page could be passed from one object to another. This is made slightly more complex by the fact that the map's functionality is being driven by Google's back end programming interface and my menu was driven by its own code library. Adding to this complexity was the fact that everything in the site was now being generated dynamically in code and nothing had a hard-coded existence that would make this a simple task. The solution came down to about ten lines of code that were the result of about four weeks of research and trial and error. It may seem like

an exorbitant amount of time to invest in such a simple piece of functionality, but I was pleased by the idea that the site had the potential to scale infinitely. In effect, I could add 10, 100, or 1000 more pages of routes (or pieces of trash) to the site and everything would continue to work without any changes to the underlying technology.

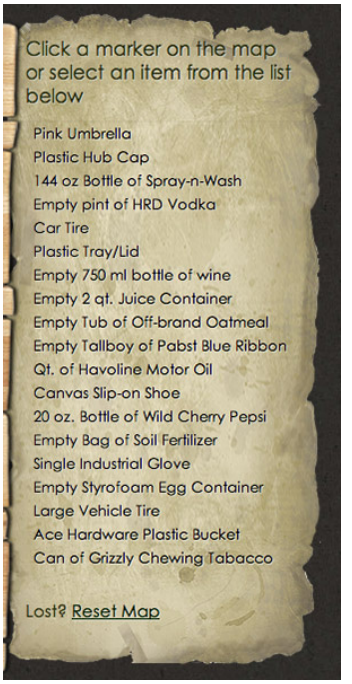


Figure 6. A screen shot of the sidebar menu from www.dottedlandscape.org.

Seeing the code behind the site work so well motivated me to think about a new addition to the site: the thumbnail view of the trash for each route sans the map interface. My students would be the first to accuse me of the sin of adding to my project's scope so late in its development — something I heavily discourage in my upper division courses — and they would be right in this critique. However, I was so encouraged by the technical developments of the project that I had to take my newfound abilities out for a spin in the name of adding a new angle to the display of the garbage. After all, the mapping functionality was the hard part. Adding a thumbnail gallery to the site would be easier to implement and, for the most part, it was.

I began building the thumbnail gallery interface by duplicating all of the site's current files and building a parallel set that simply had no maps in the main content element (an HTML 5 <article> element, in case you're wondering). I then made a copy of the JavaScript/jQuery engine that rendered the maps, markers, and sidebar menu and rewrote it to populate the now empty

map area with the thumbnails of all the images that were used in the info windows. This is a good example of how moving to the XML format allowed all the 'guts' of the site to be used in different ways with no alterations or additions to the pool of data.

Once the thumbnails were appearing, I took things one step further and added a "lightbox" effect that allows the user to call up an enlargement of the image in an overlay that floats above the page. There are dozens of code libraries that allow for this effect. However, my jQuery research led me to another add-on: the free Overlay library from jquerytools.org. [12] This particular bit of code has some nice options for styling the presentation of the overlays, including animated fade-ins, drop shadows, and embedded captions. The only downside is that all the assets for the overlay, including the enlarged images, get loaded with the page rather than when requested by the user, so the download time of the thumbnail galleries is noticeably longer than it is for the maps. The last hurdle was adapting the overlay code to the thumbnails that (like the sidebar menu) were generated dynamically in code rather than written right into the HTML, which would have been simple. Since my code engine was writing the HTML on the fly, it just took a few extra lines to embed the overlay functionality into each thumbnail. All in all, this little bit of scope creep took less than four hours to implement and, most importantly, will scale with the rest of the site if ever there is a need to add or subtract any amount of content.

In addition to the overlay plug-in, I chose to pursue another pre-built add-on for my project. The free jQuery library infobubble.js [13] can be added to a site with a Google map and lets you create highly customized info windows that can include a tabbed interface so they contain any number of extra bits of information that is easy to navigate. I used this feature to include a larger photo of each piece of trash inside of the small bit of real estate available in the pop up info window.

Figure 7. A screen shot of the tabbed info windows and custom map marker used on www.dottedlandscape.com.

Beyond the customized info windows, I also decided to create my own map markers to replace Google's default red "pin" markers. My intent was to enhance the overall quality of the site's presentation while learning even more about what was possible with Google's mapping tools. I found an online guide to creating custom markers [14] and learned how to create the markers and their separate semi-transparent shadows in a graphics editor like Photoshop. Adding these two images to the site and appending the required lines of code to the site's main code engine was a relatively simple procedure that delivered the results I was looking for.

When I set out to do this project, I knew that the technological aspects would challenge and introduce me to a new set of tools that I would have ample use for in my personal and professional pursuits. The combined use of Google's mapping programming interface, JavaScript, jQuery, and all the code libraries that complement and enhance the possibilities was indeed a valuable experience that I will continue to learn and use well into the future. With a small amount of development yet to go in the completion of the project, I anticipate even more technical challenges that I look forward to tackling.

Notes

[1] See: http://www.mapmyride.com

[2] See: http://www.csgnetwork.com/gpscoordconv.html

[3] See: http://www.itouchmap.com/latlong.html

[4] Ajax (programming). (n.d.). In *Wikipedia.* Retrieved from http://en.wikipedia.org/wiki/Ajax_(programming)

[5] XML. (n.d.). In *Wikipedia.* Retrieved from http://en.wikipedia.org/wiki/XML

[6] JSON. (n.d.). In *Wikipedia.* Retrieved from http://en.wikipedia.org/wiki/JSON

[7] Podcasting. (n.d.). In *Wikipedia.* Retrieved from http://en.wikipedia.org/wiki/Podcasting

[8] See: http://www.barebones.com/products/textwrangler

[9] See: http://jquery.com

[10] See: http://docs.jquery.com

[11] See: http://code.google.com/edu/ajax/tutorials/ajax-tutorial.html

[12] See: http://jquerytools.org/demos/overlay

[13] See: http://google-maps-utility-library-v3.googlecode.com/svn/trunk/infobubble
[14] See: www.x2tutorials.com