Technical Manual

This document provides an in depth analysis of the methodology used to create improved Google Maps content for the WAI NZ website.

FOR GOOGLE MAPS
JAVASCRIPT
WEBSITE CONTENT

TEAM RENEWEL –

BEN ALCORN

JOHN GOLDSBURY

DAMLA GUVEN

STEVEN VAN KUYK

SITAO XIE

CONTENTS

DESCRIPTION

THE PROBLEM

The old version of the Google Maps on the WAI-NZ website was the simplest implementation that the Google Maps API offers. It contained all the image data in the database, displayed with the default marker icons. All this data appeared on the map at the same time and could cause confusion and is aesthetically unappealing.

To find details about images, a window is displayed with a link to another page, navigating to this page, by default, causes users to lose the state of the map e.g. zoom level and selected marker.

Rivers and waterways are poorly defined under the road map view, and difficult to view easily on satellite view.

This map is an impractical long term solution, as the website grows attention and accumulates more data, the map will become unnavigable as markers fill the screen.

THE SOLUTION

Our team aimed to alleviate all of these issues, and create a high performance, user friendly Google Map to ensure better data portrayal and less clutter.

The main concepts in the execution of this are:

Partition map by regional council boundaries, enable users to select the region they wish to view markers in – otherwise hide markers.

Display Info Windows when markers are clicked on to show image data within the map without reloading the page itself.

Use Layers to show and hide a lake and river overlay for clarity and to inform the user of the state and location of waterways.

Remove unnecessary obfuscating elements from map views e.g. landmarks, buildings, roads.

Add marker counter to aid users in identifying the number of markers in an unselected region.

FILES AND FEATURES

SCRIPT FILES:

Listed are the JavaScript required in the display of the Google Map:

Google Maps – Provides link to Google's library server for all map interactions including the graphical user interface (GUI). Required to use the Google Maps API v3.

Map Setup – JavaScript file which makes use of Google Maps API functions to initialise and display all map content. Contains all team authored functions and variables.

Coordinates functionality – Secondary JavaScript file with functions used to store and retrieve regional council boundary coordinate data.

MapLabel: Provides functions to place text on to a map.

jQuery: Library of scripts, to be used for queries to the API when completed

These files must be referenced in whichever corresponding HTML header is used when implemented e.g. for Google Maps:

```
<script
src="http://maps.googleapis.com/maps/api/js?libraries=geometry&sens
or=false"></script>
```

And any local server files:

```
<script type="text/javascript" src="./somefile.js/"></script>
```

GEOMETRY LIBRARY:

The Geometry library provides the function:

```
google.maps.geometry.poly.containsLocation(point: latng, polygon:
Polygon)
```

Which takes a coordinate point and returns true if that point lies within a polygon.

This feature is used to detect markers within regional polygons before each is placed into its respective region.

MAP SETUP:

The file map.js contains all the functions required to produce the Google Map described in the solution.

The process that will be outlined in this document is as follows:

1. Initialise Map and global JavaScript variables
2. Fetch river and lake Polygon layers
3. Load and display regional boundary Polygons
4. Load coordinate data and create Markers
5. Create Info Windows
6. Apply event Listeners and handle user input

INITIALISATION

GLOBAL VARIABLES

The map object must be a global variable in order for the external scripts including Google Maps API to interact with the map itself.

The regions object, as well as the layer objects are referenced globally in order to respond to user interactions.

INITIAL SETUP

For a Google Map to display it must first be initialised using the '`mapOptions`' object which allows the creator to specify the general functionality of their map. This is combined with a reference to an HTML container called the '`mapCanvas`', a map will be displayed within the page as desired.

Styles are applied to the now created map, specifically many details of a default map are removed such as roads, some labels, buildings and landmarks.

Three map types are provided: Satellite, Terrain, and Simple, each have their own benefits and restrictions. Changes to the map view are handled by an event listener which will be detailed later.

The function 'initialise()' makes calls to all the functions needed to establish a working map. Each of them will be described below.

LABELS

MapLabel objects are formed during the initialisation, once the map has loaded – the text is updated to display the number of markers in each region. They are centred about the rectangular central point of a polygon.

HTML CONTROLS

A simple user interface has been provided, this consists of a dropdown menu (combo box) of regions and a button for toggling the waterways.

LAYERS

A layer is any set containing any compilation of Polygons, Points, or Polylines, and can be loaded, displayed, or hidden as a group of elements.

The river and lake polygons are displayed using fusion table layers. Fusion Tables are databases containing coordinate information and styles for polygons, and can be directly used as layers in Google Maps via the Layers interface.

Layers are shown and hidden using the '`setmap(map: …)`' function on a layer object, parsing the current map will display the layer, parsing null will hide it.

Once loaded, the layer is cached locally and will show and hide instantly. A Fusion Table is fetched using a long identifier string, this is a unique ID for each layer.

REGIONAL POLYGONS

New Zealand is divided into 16 regions, governed by corresponding Regional Councils. These regions are large and well defined, and are thus complex shapes. The current implementation contains ~6MB of coordinate data relating to the regional boundaries in the 'map-coords.js' file. Before the map is displayed, Polygon objects are created using this data, and are stored in region objects.

A region is declared as follows:

```
function region(name, polygon) {
      this.polygon = polygon;
      this.name = name;
      this.markers = [];
      this.selected = false;
      this.center = null;
      this.text = null;
}
```

Once created, a region is added to an array of regions, aptly named '`regions[]`'. These regions will be accessed in order to create and display markers as detailed below.

MARKER CREATION

Marker objects are created using the function '`createMarkers(location, iconName, info)`' which takes a location in the form of a latlng object, iconName – the filename of the marker image, and info – the pre initialised info window.

For each region, if a location is within the polygon pertaining to that regional boundary, a marker will be created and added to region.markers using the '`regions[i].markers.push(marker)`' method.

Upon creation, hide this marker in preparation for user interaction.

INFO WINDOWS

Applied to each marker, is an Info Window which contains image and other related data. The content of an Info Window is HTML with references to JavaScript variables.

An info window is created using the function:

```
createInfoWindow(fileName, title, description, date, tags)
```

Each of these arguments are provided by server data and refer to static server images.

An API is in development to pull this data from the web server, but the current solution uses a hardcoded array called '`points[]`'.

A prototype for using jQuery to fetch a JSON file from the proposed API is located in the initialisation function.

EVENT HANDLING AND LISTENERS

CLICKS

Mouse clicks are handled differently for clicks on Regional polygons and Markers -

Markers:

In the initialisation stage, every marker has a click listener applied to it.

Upon detecting a click, the marker will display its corresponding info window.

If another marker is clicked while another's info window is open, the previous info window will close and the current will open.

Regions:

Region polygons have three event listeners applied: click, mouseover, and mouseout.

When clicked, a region will become selected and toggle between a state of showing markers and hiding markers. Additionally, when selected, the marker counter text is hidden until the region is deselected.

When showing markers, the transparent colour is removed and the outline of the region is thickened. When hiding markers, the outline is thinned and the transparent colour is reapplied.

The colour scheme for the regional polygons is set in global variables at the top of the JavaScript file.

If a region with an info window is deselected, the info window will close.

MOUSEOVERS AND MOUSEOUTS

If a cursor enters the boundary of a region, that region will become more opaque in order to encourage user interaction. When it exits, the colour returns to a more transparent colour.

TOGGLE WATERWAYS

When the 'Toggle Waterways' button is clicked, a script in the HTML file will trigger the '`displayRivers()`' function, showing or hiding the rivers.

When first pressed, the data must stream from the FusionTable, afterwards interactions are cached.

REGION COMBO-BOX

When a user selects an option in the combo box (located in the top right hand corner of the map), the function '`regionSelector()`' is executed. If "Show All" is selected, every region becomes selected and all markers will be shown. Contrastingly, if "Hide All" is selected, each region becomes unselected and all markers will be hidden (and info windows closed). If a region is seleceted, the name of the region is gathered from the combo box using '`document.getElementById("region").value`'. '`Regions[]`' is then iterated through until a matching name is found.

When the combo box is used, if any other regions are selected, they will be deselected, leaving only the region chosen to be selected.