Project Background

The Google Maps interface has become a very popular tool on the Internet in the last couple of years. Google Map allows Internet users to explore locations on a world map, look up addresses and get driving directions from one specified location to another. Google also offers an API that allows developers to implement their own solutions using the Google Maps technology. The availability of the API has been a huge hit with developers, however the average Internet user does not posses the skills needed to build their own applications. The aim of this project is to remove the steep learning curve associated with the API. With this tool, the user only needs to learn a simple specification language designed to customized Google Maps. Once the input specification language is compiled by ELI, the end product is a HTML page with tabs each containing a Google Map with features specified by the user.

Language Description

The specification language is built to be as simple as possible and to use key words that are intuitive. We list below examples that reflect features of the language.

A user can define a point which is an abstract location on the globe. This point can later be used to specify a map's center or a marker's position. The general pattern to declare a point is:

Create Point PointName(longitude, latitude);

Longitude and latitude specify the point's coordinates.

The general pattern to declare a new map is:

Create Map MapName with ListOfAttributes;

The statement declares a new Google Map to be created inside a tab on the existing webpage. ListOfAttributes is a list of attributes for the map. Possible attributes are:

- Name: sets the tab name.
- View: defines the initial map view (map, satellite or hybrid)
- Location: sets the location on which the map is centered
- Zoom: sets the initial zoom of the map

An example of the Create Map feature is:

Create Map BoulderMap with Name “Boulder, CO”, Location (-105, 40), View map;

If a certain attribute is not defined for a map, a default will be set.

A marker is an icon that can appear at a single point on a map.
The general pattern to declare a new marker is:

**Create Marker** MarkerName with ListOfAttributes;

**ListOfAttributes** is a list of attributes for the marker. Possible attributes are:
- Name: sets the name to be displayed in the marker's info window
- Location: sets the location at which the marker is to be placed
- Info: defines the information to be displayed in the info window

An example of the Create Marker feature is:

Create Marker MyHouse with Name “Brice's House”, Location BricePoint, Info “this is were I live!”;

Here BricePoint refers to a point that has been previously declared with a **Create Point** statement.

A trace is composed of multiple sections. Each section is a line defined by a start point and an end point.
The general pattern to define a trace is:

**Create Trace** TraceName(ListOfPoints);

**ListOfPoints** is a list of previously declared points.

An example of the Create Trace feature is:

Create Trace pathToMyHouse(P1, P2);

The general pattern to place a marker on a map is:

**Place** MarkerName on ListofMaps;

**ListofMaps** is a list of previously declared maps to place the marker on.

An example of the Place features is:

Place MyHouse on BoulderMap, M1, M2, M3;

The general pattern to place a trace on a map is:

**DrawTrace** TraceName on ListofMaps;
At any time, valid attributes of a Map, Marker or a Trace can be reset with the `Set` command. For example, to set the zoom level and name of a map, a user would put:

```
Set BoulderMap with Zoom 3, Name “Denver”;
```

The specification language also allows for various scopes. For example, the following statements are allowed:

```
Create Map BoulderMap with Location P1;
Create Marker M1 with Location P1;
Create Marker M2 with Location P2;
For WorkSheet1
    Create Marker M1 with Location P2;
    Place M1 on BoulderMap;
    Place M2 on BoulderMap;
End;
```

A work sheet has similar scoping rule as a block statement in C. In the above example, the applied occurrence of M1 in WorkSheet1 refers to the defining occurrence in the worksheet, whereas the applied occurrence of M2 refers to the defining occurrence above the `For` statement.

**Requirements**

**Source to Source**
The input source is the specification language that we have just defined. The output source is HTML containing JAVASCRIPT elements.

**Name analysis**
Entities are created using `Create` statements. The source language uses C scoping rules. A worksheet is similar to a C block and inherits the same scope attributes.

**Structured Output**
The output will be produced using PTG.