Flight Integration

**Basic Procedure**

1. Obtain data from via APIs from any flight company
2. Set up a http server and store any files needed in the webroot directory so they can be retrieved via localhost
3. Parse the data with a JSON parser to convert the string in JSON format and store it in a php file
4. Read the data and use variables to store any information needed
5. Download Micello’s example on custom menu pop up links

(Refer to documentation for a better grasp on how the code should look like)

1. Edit the load community and pop-ups so that they are not hard-coded to a single geometry id
2. Set up a variable to store the clicked id
3. Using the clicked id, print out the information you want to display in the custom menu pop up link

**Documentation**

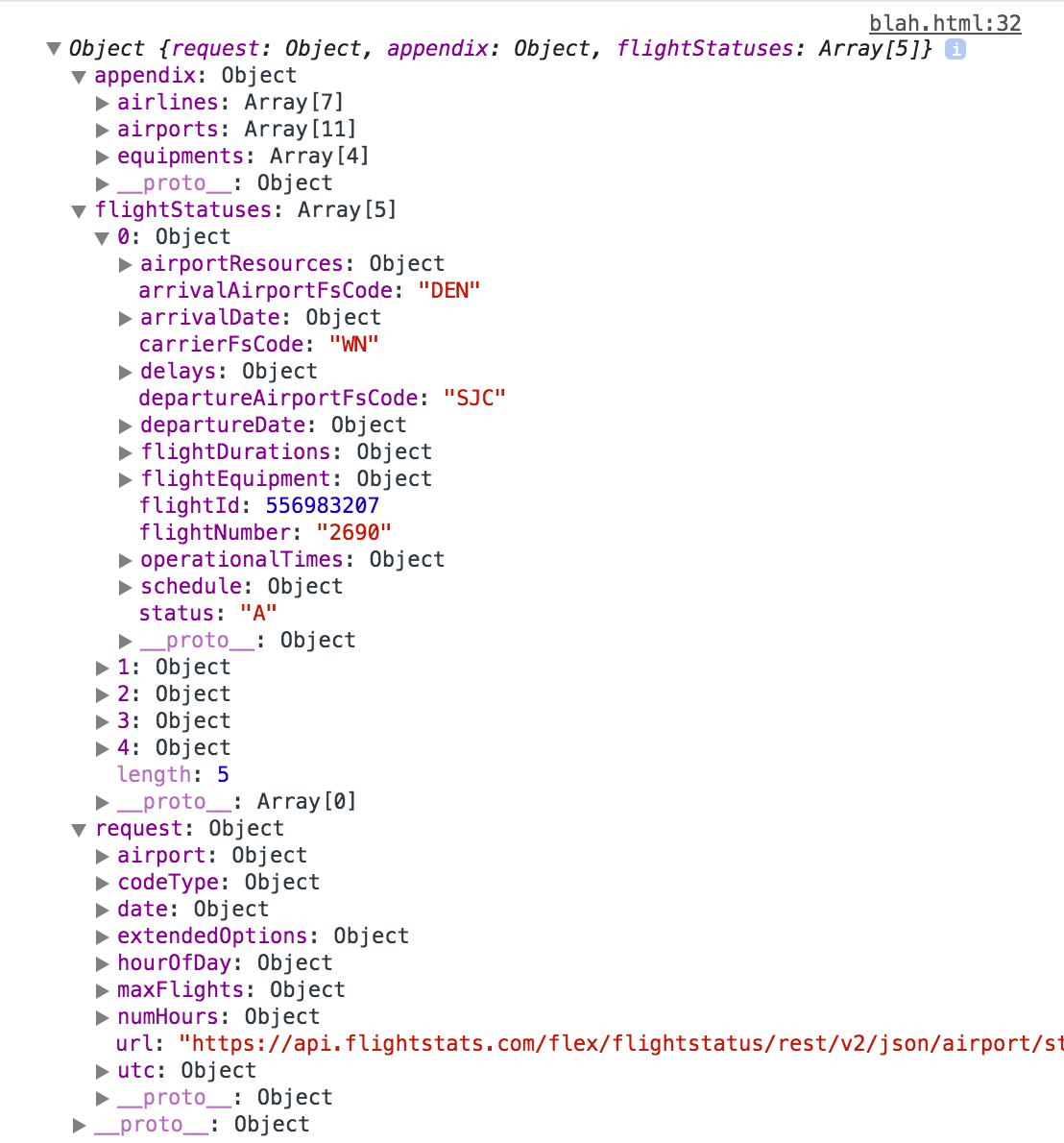
In order to start this integration, you must first be able to find an api containing flight data. For this specific example, we decided to use Flightstats api and pull information about flights that depart from the San Jose International Airport at 2pm (14:00 hours) everyday. (Meaning the data is set to the current date) In this example, we only incorporated 5 flights and put them on the maps.

In this demo, through the use of an http server and the mapClick override event, we are able to access the data from Flightstats and create a custom popup to represent the data. Each of the files created should be stored in the webroot directory of the server, so you can call it using localhost.

Upon retrieving the data (for flightstats, we were given a url to be curled), we have to use a JSON parser in order to convert string to JSON format . I created a php file in order to make a webservice call to the url given to me by Flightstats. Currently, it would look like this:



With the parser, we can get it to look like this:



You can expand or collapse the object, but the main point is that now we can access all the data we need by parsing the string. In this example, using my object that was parsed is named **det**. By calling **det.flightStatuses[0].flightNumber**, I can access the flightNumber of the first object in the array under flightStatuses. **det.flightStatues[1].flightNumber** would access the second flightNumber and so on. Now that we know how to access the data, it’s time to pull it up on the maps.

By using the documentation from the **Custom menu link in popup** page, we are able to start off with a basic custom menu link. However, since that load community is set to Hillsdale Shopping Center, we want to change the load community to 144 (San Jose International Airport). Since these menu links are hardcoded to specific id’s, we want to display the custom link in the id that we click. We can do this by storing the click.id into a variable which we can call on later to use in the popups.

Here is the code used to create a custom pop-up and be able to hide it right after:

**mapControl.hideInfoWindow();**

**if(mapControl.popupFlags) {**

**mapControl.showInfoWindow(mapDataObject.geomMap[id].g, html);**

**}**

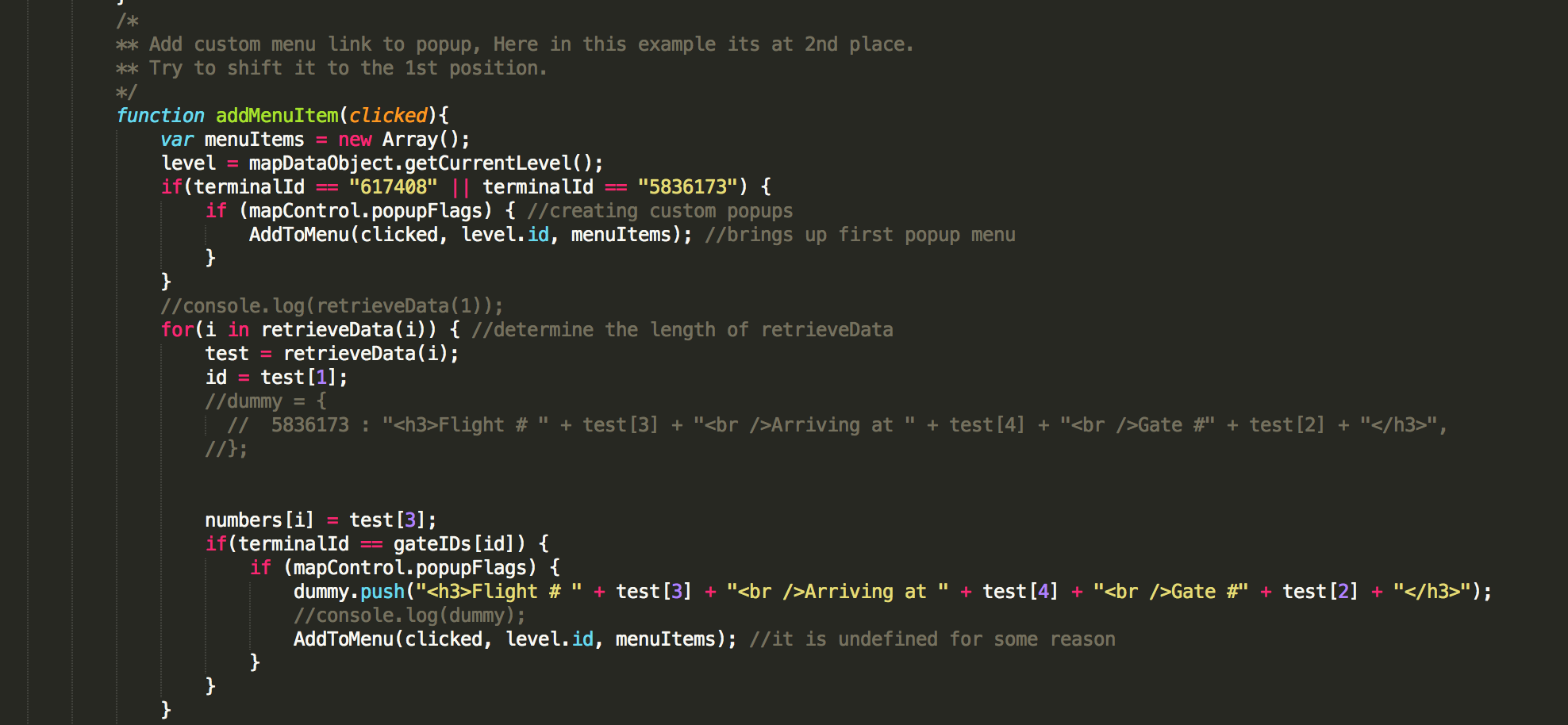
where “id” is the click.id of the geometry(or whatever id you want it to be) and html is the data/string/information that you want to be displayed.

The next step is after setting up the custom pop-up, we must push it to **menuItems** so that it can be added through the **AddToMenu** function. Here is an example of what the entire thing should look together:

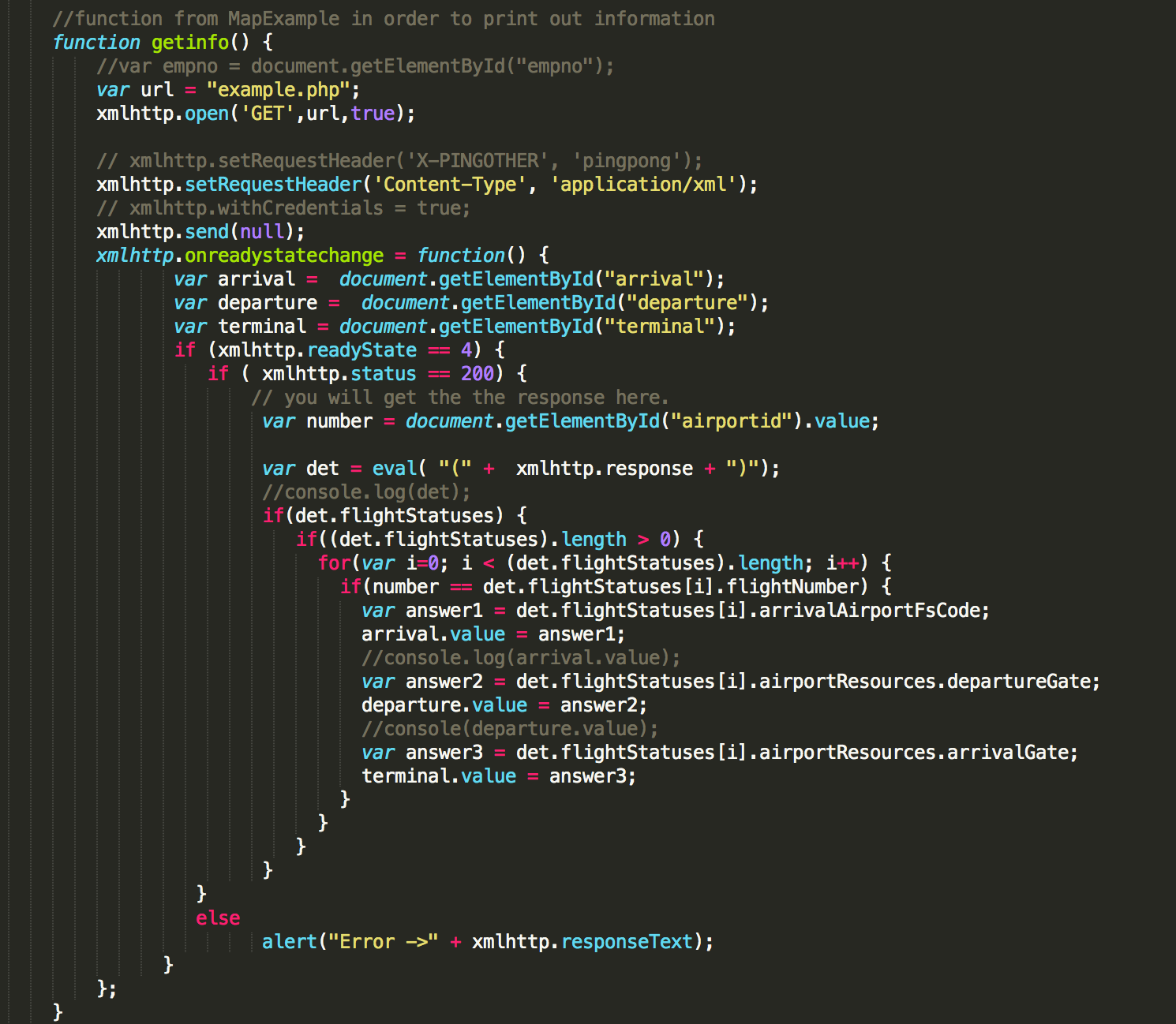


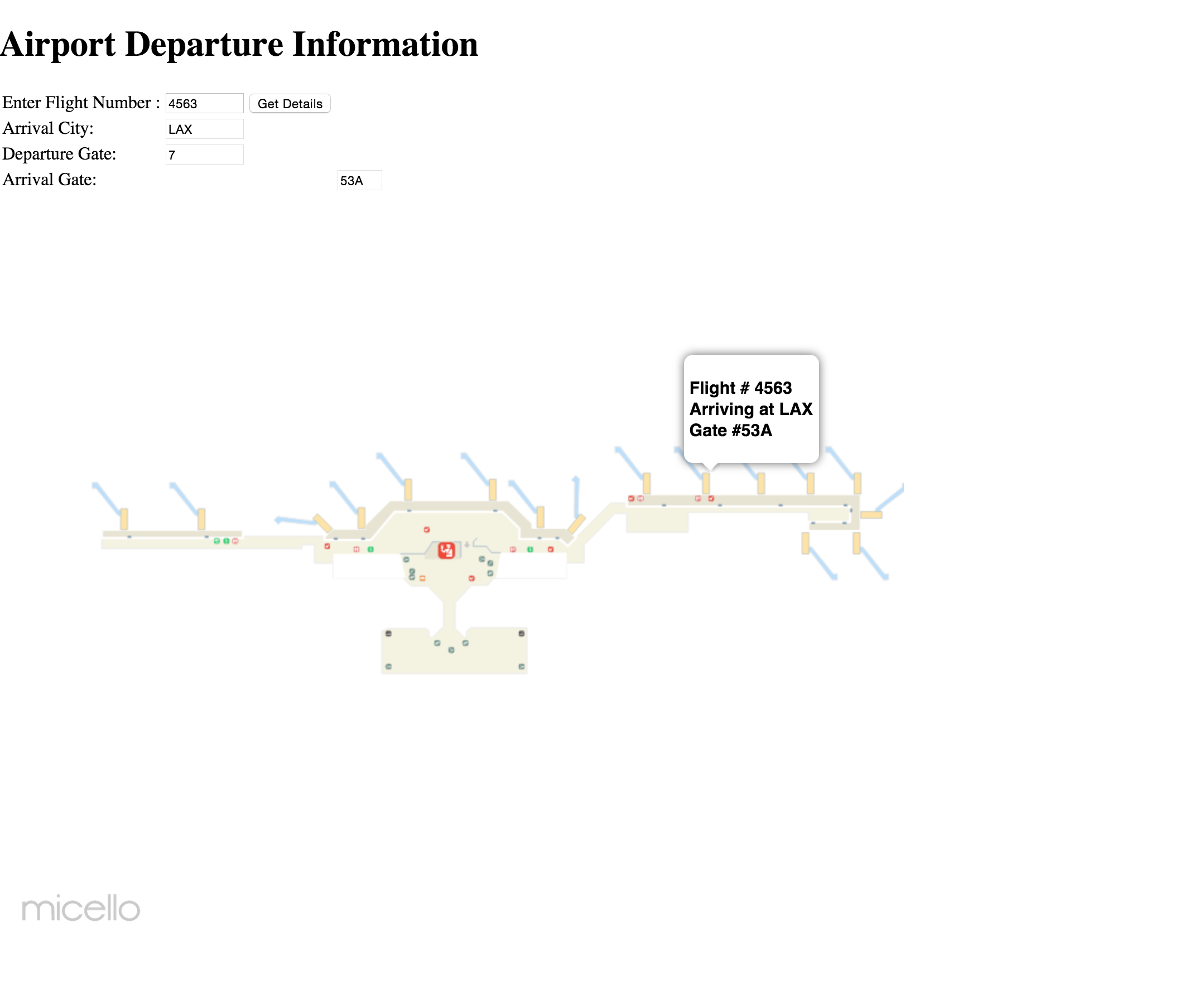
Flights is the variable we want to add to our menu via custom menu popups and we do that by pushing it into the object array menuItems. **Dummy** is the variable used to show the flight information in the example above.

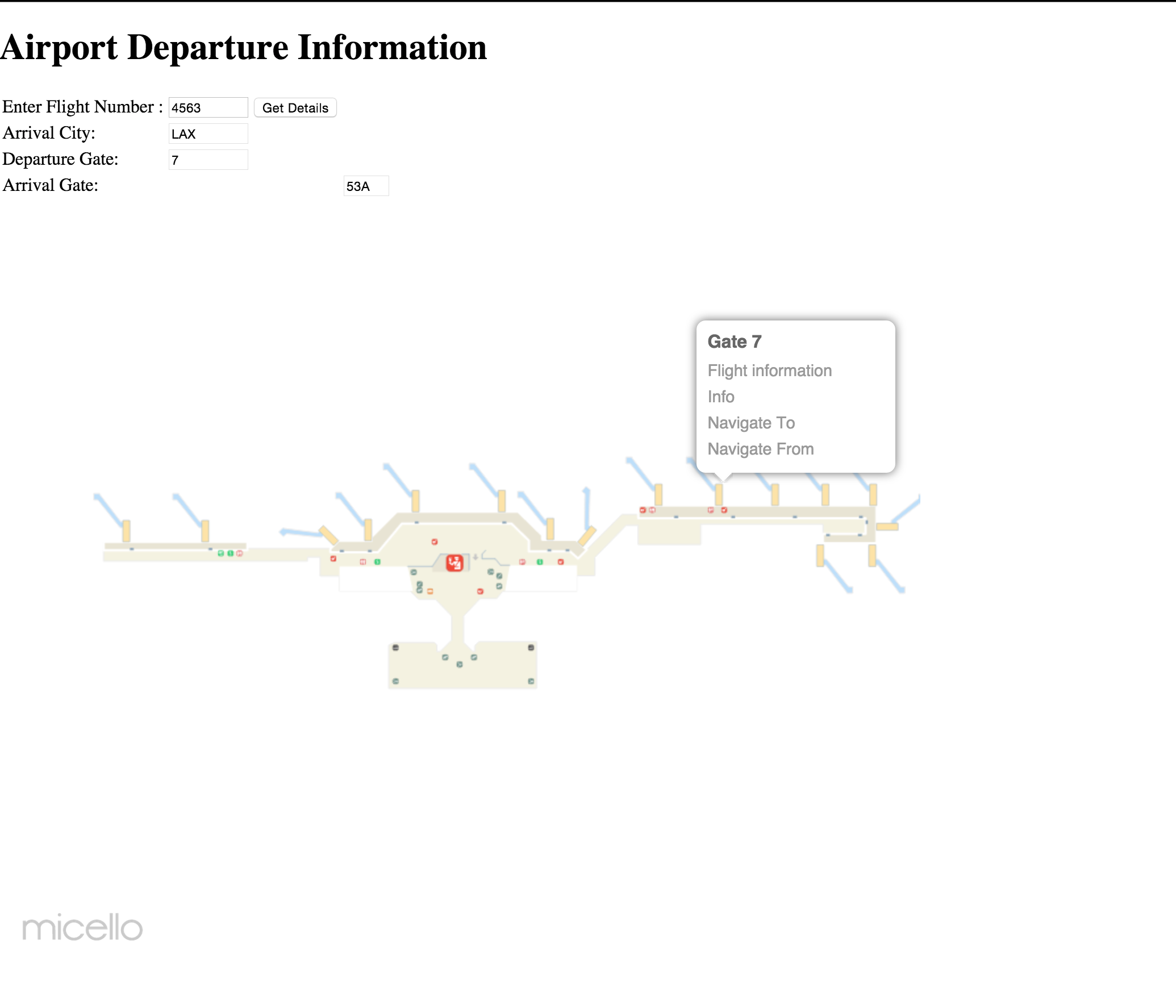
The next step is adding the specific thing we want to display to the id that wants to be displayed. By using the flight data, we were able to access which gates the flight would departure at, so we used those gates as our id to show the flights. By saving the departure gates in array, we can call it in the **addMenuItem** function and see if the gateID equals the clicked.id. If it does, then we can **AddToMenu** the variable we want to show in the popup. See example below:



By doing this, the pop-ups of the flight information and which gate they pertain too should be visible on the map itself.

The final step of the integration would be added in Airport Departure Information so it would be easier to tell which gate is which. By knowing the flight number, you can find out the Arrival City, Departure Gate, and Arrival Gate as well as where to go in SJC. The working code is shown to the left.

Final Product:



**Procedure**

1. Obtain data from via APIs from any flight company
2. Set up a http server and store any files needed in the webroot directory so they can be retrieved via localhost
3. Parse the data with a JSON parser to convert the string in JSON format and store it in a php file
4. Read the data and use variables to store any information needed
5. Download Micello’s example on custom menu pop up links
6. Edit the load community and pop-ups so that they are not hard-coded to a single geometry id
7. Set up a variable to store the clicked id
8. Using the clicked id, print out the information you want to display in the custom menu pop up link