Data Layer: Drag and Drop GeoJSON

To use this example, drag and drop a GeoJSON file or files from your computer on to the map above. You can also drag text or HTML GeoJSON content from another website or from the sample below.

View this example full screen or check out the reference documentation for google.maps.Data.

JavaScript | JavaScript + HTML | Sample GeoJSON

```
/* Map functions */

var map;

function initMap() {
  // set up the map
  map = new google.maps.Map(document.getElementById('map-canvas'), {
    center: new google.maps.LatLng(0, 0),
    zoom: 2
  });
}
```

```
function loadGeoJsonString(geoString) {
  var geojson = JSON.parse(geoString);
  map.data.addGeoJson(geojson);
  zoom(map);
}

/**
 * Update a map's viewport to fit each geometry in a dataset
 * @param {google.maps.Map} map The map to adjust
 */
function zoom(map) {
  var bounds = new google.maps.LatLngBounds();
  map.data.forEach(function(feature) {
    processPoints(feature.getGeometry(), bounds.extend, bounds);
  });
  map.fitBounds(bounds);
}

/**
 * Process each point in a Geometry, regardless of how deep the points may lie.
 * @param {google.maps.Data.Geometry} geometry The structure to process
 * @param {function(google.maps.LatLng)} callback A function to call on each
 *     LatLng point encountered (e.g. Array.push)
 * @param {Object} thisArg The value of 'this' as provided to 'callback' (e.g.
 *     myArray)
 */
function processPoints(geometry, callback, thisArg) {
  if (geometry instanceof google.maps.LatLng) {
    callback.call(thisArg, geometry);
  } else if (geometry instanceof google.maps.Data.Point) {
    callback.call(thisArg, geometry.get());
  } else {
    geometry.getArray().forEach(function(g) {
      processPoints(g, callback, thisArg);
    });
  }
}


/* DOM (drag/drop) functions */

function initEvents() {
  // set up the drag & drop events
  var mapContainer = document.getElementById('map-canvas');
  var dropContainer = document.getElementById('drop-container');

  // first on common events
  [mapContainer, dropContainer].forEach(function(container) {
    container.addEventListener('drop', handleDrop, false);
    container.addEventListener('dragover', showPanel, false);
  });

  // then map-specific events
  mapContainer.addEventListener('dragstart', showPanel, false);
  mapContainer.addEventListener('dragenter', showPanel, false);

  // then the overlay specific events (since it only appears once drag starts)
```

```
  dropContainer.addEventListener('dragend', hidePanel, false);
  dropContainer.addEventListener('dragleave', hidePanel, false);
}

function showPanel(e) {
  e.stopPropagation();
  e.preventDefault();
  document.getElementById('drop-container').style.display = 'block';
  return false;
}

function hidePanel(e) {
  document.getElementById('drop-container').style.display = 'none';
}

function handleDrop(e) {
  e.preventDefault();
  e.stopPropagation();
  hidePanel(e);

  var files = e.dataTransfer.files;
  if (files.length) {
    // process file(s) being dropped
    // grab the file data from each file
    for (var i = 0, file; file = files[i]; i++) {
      var reader = new FileReader();
      reader.onload = function(e) {
        loadGeoJsonString(e.target.result);
      };
      reader.onerror = function(e) {
        console.error('reading failed');
      };
      reader.readAsText(file);
    }
  } else {
    // process non-file (e.g. text or html) content being dropped
    // grab the plain text version of the data
    var plainText = e.dataTransfer.getData('text/plain');
    if (plainText) {
      loadGeoJsonString(plainText);
    }
  }

  // prevent drag event from bubbling further
  return false;
}


google.maps.event.addDomListener(window, 'load', function() {
  initMap();
  initEvents();
});
```

Last updated March 17, 2015.