Mapping User Submitted Data
LA 558 Final Project

Bailey Hanson

Project Goals

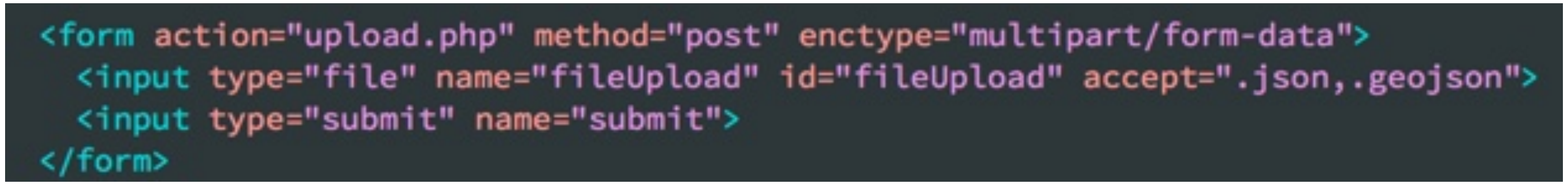
- Allow users to upload their own GeoJSON data to a Leaflet map.
- Use Turf.js to provide analysis tools (center, convex, simplify, area, and lineDistance).
- Allow the use of analysis tools based on the uploaded GeoJSON geometry type (Point, Polygon, LineString). For example, only area calculations would be provided for Polygon GeoJSON layers.
- Use HTML, CSS, Leaflet, Turf.js, JSON/GeoJSON, jQuery, PHP, SQL

Methodology

1. HTML page for the file upload
2. Upload.php file, and an upload directory on the server
3. SQL table in MariaDB Database
4. PHP page to query file from Database
5. Functions to run Turf.js processes on uploaded data

Methodology - HTML page

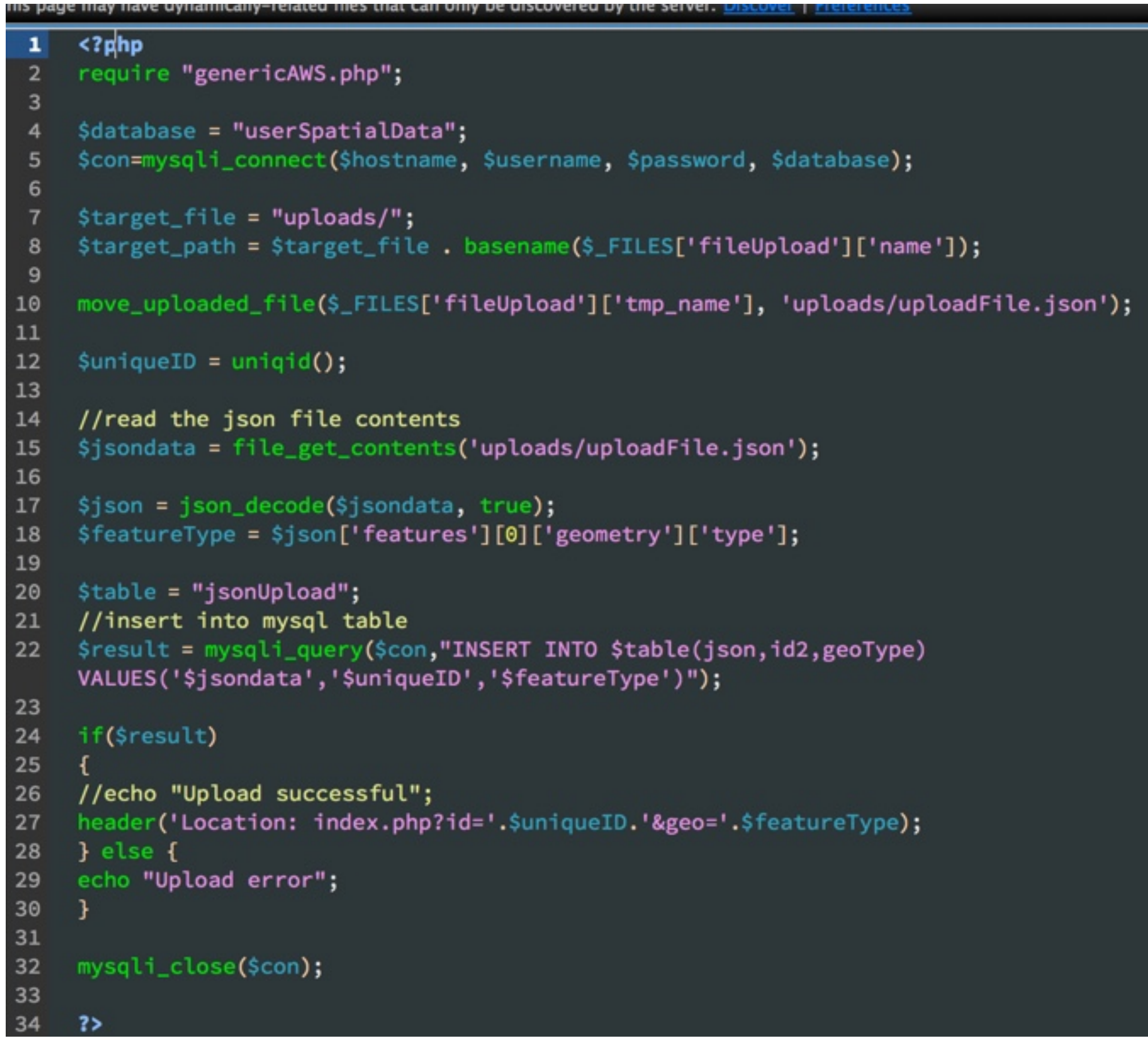
- Contains an HTML input form that accepts .json and .geojson file extensions and loads the upload.php file through the post method.



```
<form action="upload.php" method="post" enctype="multipart/form-data">
  <input type="file" name="fileUpload" id="fileUpload" accept=".json,.geojson">
  <input type="submit" name="submit">
</form>
```

Methodology – Upload.php

- Database connection information
- Path the save the GeoJSON file on the server for reading
- Save the contents of the GeoJSON file as a php variable
- Extract the geometry type from the php variable
- SQL insert using php variables
- Load index.php and pass uniqueID and geometry type in the URL
- Close SQL connection

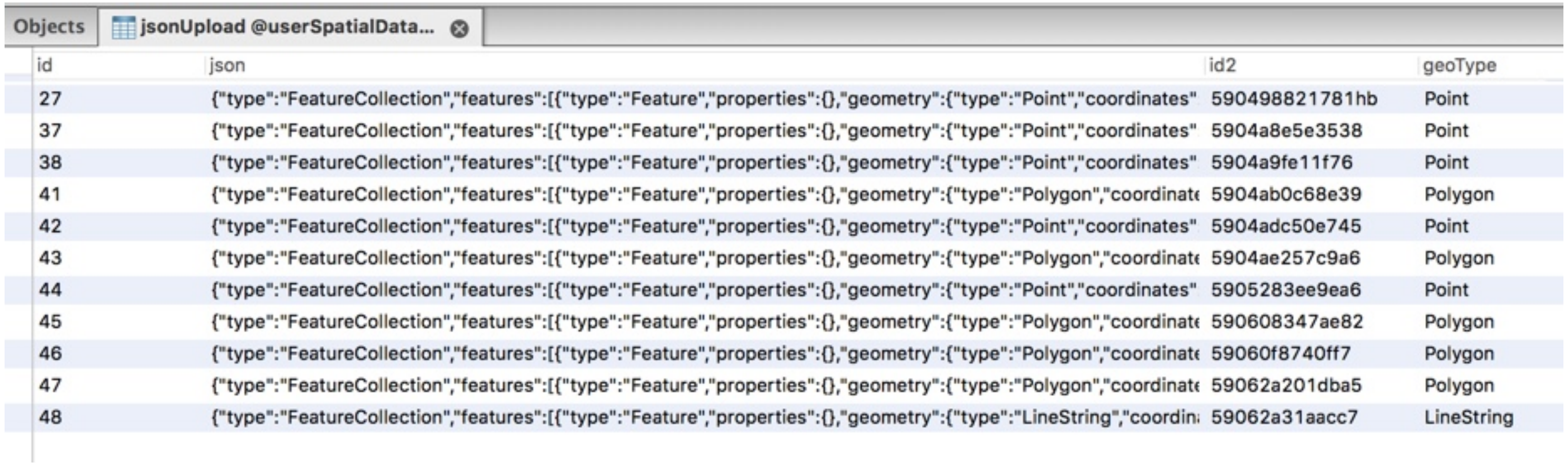


his page may have dynamically-related files that can only be discovered by the server. Discover | Preferences

```php
<?php
require "genericAWS.php";

$database = "userSpatialData";
$con=mysqli_connect($hostname, $username, $password, $database);

$target_file = "uploads/";
$target_path = $target_file . basename($_FILES['fileUpload']['name']);

move_uploaded_file($_FILES['fileUpload']['tmp_name'], 'uploads/uploadFile.json');

$uniqueID = uniqid();

//read the json file contents
$jsondata = file_get_contents('uploads/uploadFile.json');

$json = json_decode($jsondata, true);
$featureType = $json['features'][0]['geometry']['type'];

$table = "jsonUpload";
//insert into mysql table
$result = mysqli_query($con,"INSERT INTO $table(json,id2,geoType)
VALUES('$jsondata','$uniqueID','$featureType')");

if($result)
{
//echo "Upload successful";
header('Location: index.php?id='.$uniqueID.'&geo='.$featureType);
} else {
echo "Upload error";
}

mysqli_close($con);

?>
```

Methodology - SQL table in MariaDB Database

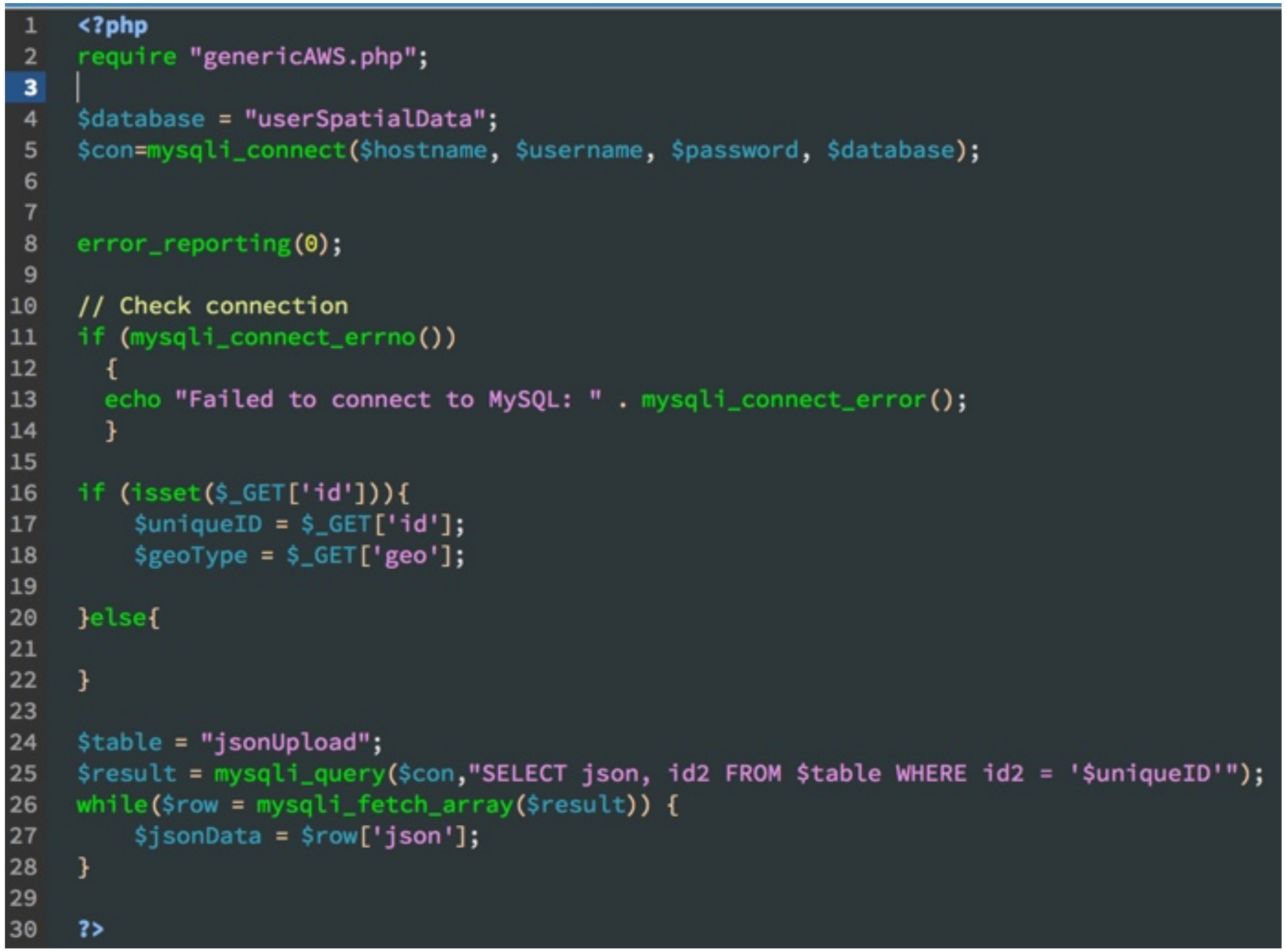
- The SQL table has four fields
  - Id – record id
  - Json – the contents of the GeoJSON file
  - Id2 – unique id created by the upload.php file
  - geoType – stores the geometry type of the GeoJSON file uploaded



id	json	id2	geoType
27	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Point","coordinates"	590498821781hb	Point
37	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Point","coordinates"	5904a8e5e3538	Point
38	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Point","coordinates"	5904a9fe11f76	Point
41	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Polygon","coordinat	5904ab0c68e39	Polygon
42	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Point","coordinates"	5904adc50e745	Point
43	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Polygon","coordinat	5904ae257c9a6	Polygon
44	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Point","coordinates"	5905283ee9ea6	Point
45	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Polygon","coordinat	590608347ae82	Polygon
46	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Polygon","coordinat	59060f8740ff7	Polygon
47	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"Polygon","coordinat	59062a201dba5	Polygon
48	{"type":"FeatureCollection","features":[{"type":"Feature","properties":{},"geometry":{"type":"LineString","coordin	59062a31aacc7	LineString

Methodology – index.php – PHP Code

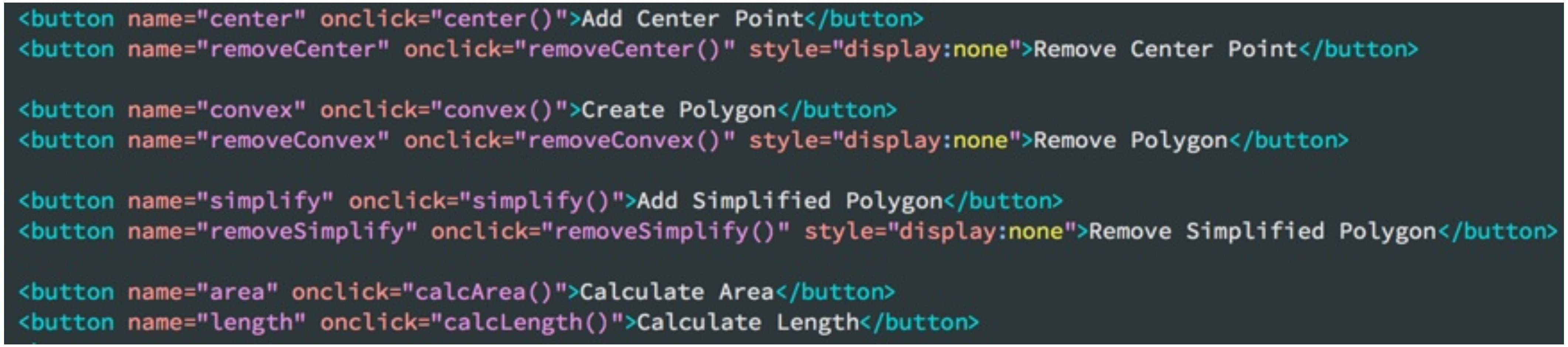
- Database connection information
- $_GET variables id and geo passed from the upload.php
- SQL query to get the json field in the database based on unique id passed from upload php



```php
<?php
require "genericAWS.php";

$database = "userSpatialData";
$con=mysqli_connect($hostname, $username, $password, $database);


error_reporting(0);

// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

if (isset($_GET['id'])){
    $uniqueID = $_GET['id'];
    $geoType = $_GET['geo'];

}else{

}

$table = "jsonUpload";
$result = mysqli_query($con,"SELECT json, id2 FROM $table WHERE id2 = '$uniqueID'");
while($row = mysqli_fetch_array($result)) {
    $jsonData = $row['json'];
}

?>
```

Methodology – index.php – HTML Code

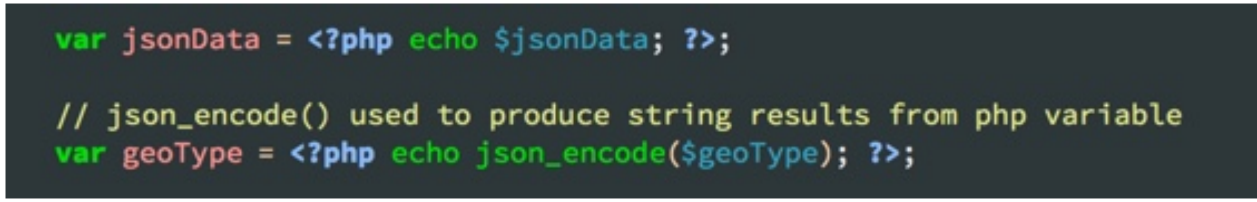
- Standard HTML elements (script source links, css style, paragraph text, input form, and buttons)
- Each button is named, runs a function on click (Turf.js analysis), and some have display setting set.



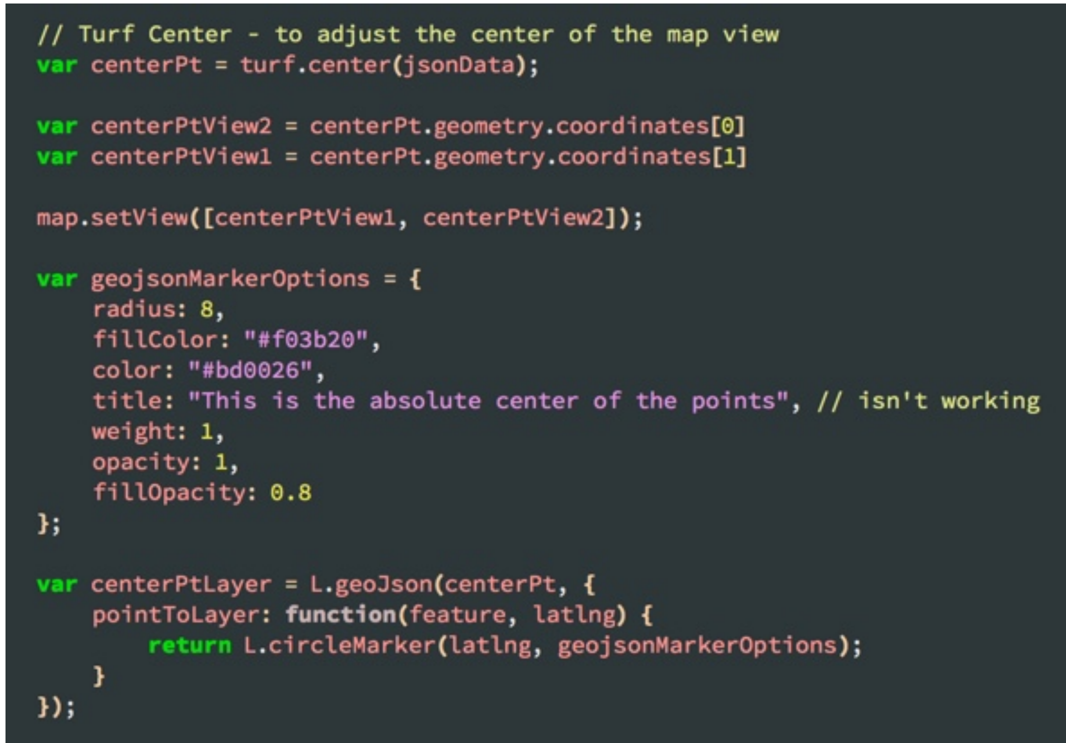
```html
<button name="center" onclick="center()">Add Center Point</button>
<button name="removeCenter" onclick="removeCenter()" style="display:none">Remove Center Point</button>

<button name="convex" onclick="convex()">Create Polygon</button>
<button name="removeConvex" onclick="removeConvex()" style="display:none">Remove Polygon</button>

<button name="simplify" onclick="simplify()">Add Simplified Polygon</button>
<button name="removeSimplify" onclick="removeSimplify()" style="display:none">Remove Simplified Polygon</button>

<button name="area" onclick="calcArea()">Calculate Area</button>
<button name="length" onclick="calcLength()">Calculate Length</button>
```

Methodology – index.php – JavaScript Code

- Standard Leaflet code
- PHP variable echoed to JavaScript variable (GeoJSON data, and geotype)
- json_encode() used to set geoType to string
- Turf.center to find absolute center for uploaded data, then used to setView of map
- Variables for circle marker style settings, and centerPtLayer (used by functions onclick)



```
var jsonData = <?php echo $jsonData; ?>;

// json_encode() used to produce string results from php variable
var geoType = <?php echo json_encode($geoType); ?>;
```

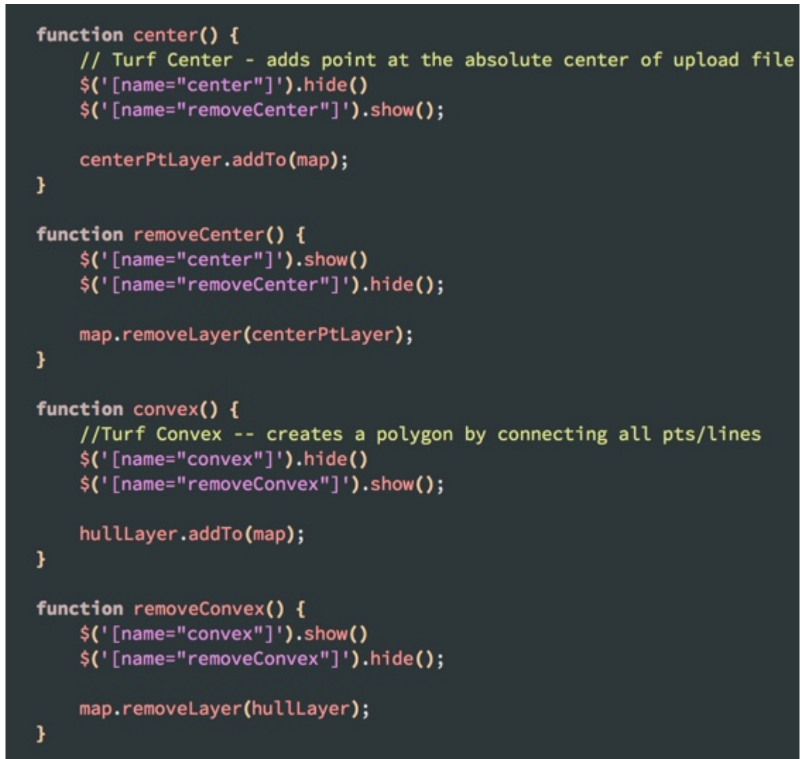


```
// Turf Center - to adjust the center of the map view
var centerPt = turf.center(jsonData);

var centerPtView2 = centerPt.geometry.coordinates[0]
var centerPtView1 = centerPt.geometry.coordinates[1]

map.setView([centerPtView1, centerPtView2]);

var geojsonMarkerOptions = {
    radius: 8,
    fillColor: "#f03b20",
    color: "#bd0026",
    title: "This is the absolute center of the points", // isn't working
    weight: 1,
    opacity: 1,
    fillOpacity: 0.8
};

var centerPtLayer = L.geoJson(centerPt, {
    pointToLayer: function(feature, latlng) {
        return L.circleMarker(latlng, geojsonMarkerOptions);
    }
});
```

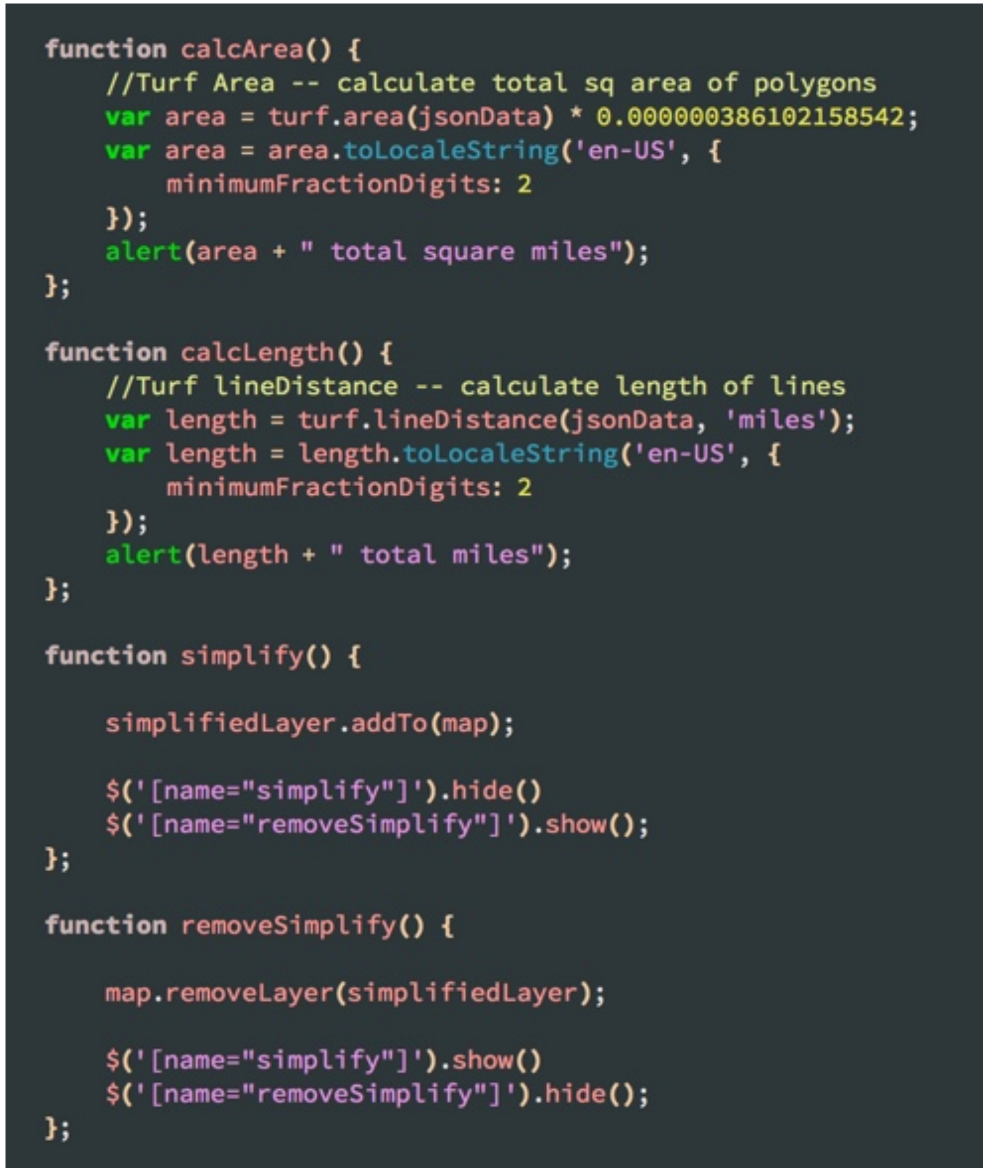
Methodology – index.php – JavaScript Code

Functions: used to add and remove layers produced through Turf.js (and buttons)

- center()
- removeCenter()
- convex()
- removeConvex()
- calcArea()
- calcLength()
- simplify()
- removeSimplify()



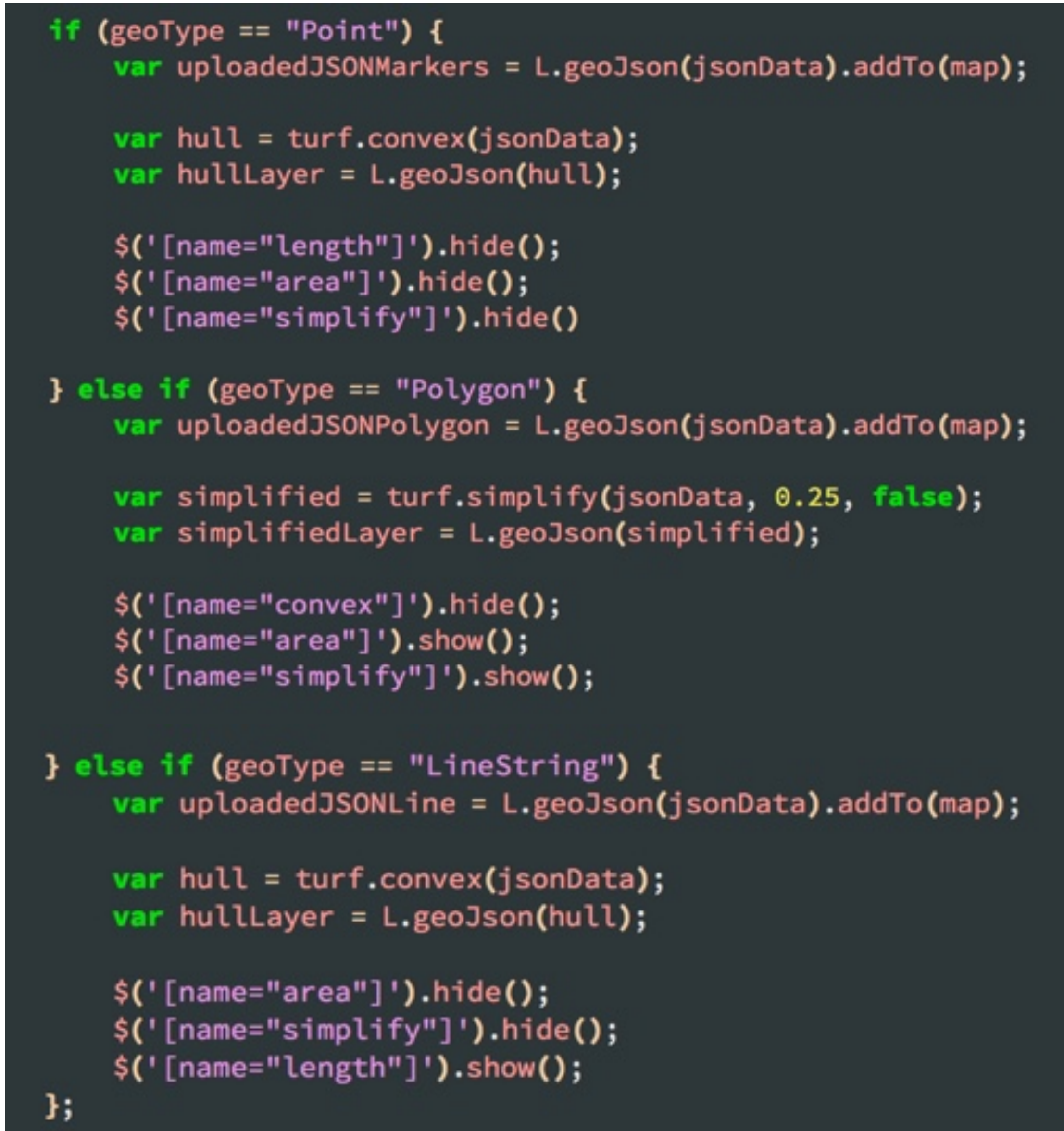
```
function center() {
    // Turf Center - adds point at the absolute center of upload file
    $('[name="center"]').hide()
    $('[name="removeCenter"]').show();

    centerPtLayer.addTo(map);
}

function removeCenter() {
    $('[name="center"]').show()
    $('[name="removeCenter"]').hide();

    map.removeLayer(centerPtLayer);
}

function convex() {
    //Turf Convex -- creates a polygon by connecting all pts/lines
    $('[name="convex"]').hide()
    $('[name="removeConvex"]').show();

    hullLayer.addTo(map);
}

function removeConvex() {
    $('[name="convex"]').show()
    $('[name="removeConvex"]').hide();

    map.removeLayer(hullLayer);
}
```



```
function calcArea() {
    //Turf Area -- calculate total sq area of polygons
    var area = turf.area(jsonData) * 0.000000386102158542;
    var area = area.toLocaleString('en-US', {
        minimumFractionDigits: 2
    });
    alert(area + " total square miles");
};

function calcLength() {
    //Turf lineDistance -- calculate length of lines
    var length = turf.lineDistance(jsonData, 'miles');
    var length = length.toLocaleString('en-US', {
        minimumFractionDigits: 2
    });
    alert(length + " total miles");
};

function simplify() {

    simplifiedLayer.addTo(map);

    $('[name="simplify"]').hide()
    $('[name="removeSimplify"]').show();
};

function removeSimplify() {

    map.removeLayer(simplifiedLayer);

    $('[name="simplify"]').show()
    $('[name="removeSimplify"]').hide();
};
```

Methodology – index.php – JavaScript Code

- If else statement based on uploaded GeoJSON geometry type
  - Turf.js code
  - Hide/show buttons



```
if (geoType == "Point") {
    var uploadedJSONMarkers = L.geoJson(jsonData).addTo(map);

    var hull = turf.convex(jsonData);
    var hullLayer = L.geoJson(hull);

    $('[name="length"]').hide();
    $('[name="area"]').hide();
    $('[name="simplify"]').hide()

} else if (geoType == "Polygon") {
    var uploadedJSONPolygon = L.geoJson(jsonData).addTo(map);

    var simplified = turf.simplify(jsonData, 0.25, false);
    var simplifiedLayer = L.geoJson(simplified);

    $('[name="convex"]').hide();
    $('[name="area"]').show();
    $('[name="simplify"]').show();


} else if (geoType == "LineString") {
    var uploadedJSONLine = L.geoJson(jsonData).addTo(map);

    var hull = turf.convex(jsonData);
    var hullLayer = L.geoJson(hull);

    $('[name="area"]').hide();
    $('[name="simplify"]').hide();
    $('[name="length"]').show();
};
```

Conclusions

- Useful tool to upload GeoJSON formatted data and run specific analysis functions based on Geometry type
- Future improvements
  - Error alert for uploading invalid GeoJSON data
  - Combine JSON data to GeoJSON features (example: upload population data in JSON format and combine with GeoJSON to show choropleth maps)
  - Upload two GeoJSON files and run tools that use both (example: upload lines and polygons and slip polygons based on the lines)