File - C:\wamp64\www\wrestoreBCK2\model\js\mapping_new.js

```
// JavaScript Document
var colors = ['#FF0000', '#00FF00', '#0000FF', '#FFFF00'];
var backArray = [];
// var backArray2 = [];
var cropArray = [];
// var cropArray2 = [];
var stripArray = [];
// var stripArray2 = [];
var wetArray = [];
// var wetArray2 = [];
var conserveArray = [];
// var conserveArray2 = [];
var grassArray = [];
// var grassArray2 = [];
var filterArray = [];
// var filterArray2 = [];
var coverArray = [];
// var coverArray2 = [];
//var listAll;
var number;
var map;
var map1;
// var map2;
var grass;
var wetlands;
// var wetlands2;
var filter;
// var filter2;
var strip;
// var strip2;
var notill;
var cover;
var background;
// var background2;
var crop;
// var crop2;
var filterColor;
// var blueIcon = "http://wrestore.iupui.edu/model/images/green.png";
// var starIcon = "http://wrestore.iupui.edu/model/images/gw.png";
var tillIcon = "http://wrestore.iupui.edu/model/images/noTill.png";
var wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands.png";
var grassIcon = "http://wrestore.iupui.edu/model/images/grass.png";
var wetlandsize;
var filterAcre;
// var size;
// var obj;
//var obj1;
var lots = [];
// This is firing off around 800 of tools.php

function initialize() {

    //empty the div that shows all the data if they click on a sub basin
    $('#tools').empty();
    $('#toolpic').empty();
    //$('#tools').append('Click anywhere on this yellow bar for additional info on the icons representing these practices<br
/>');

    // ////// START Drawing MAIN MAP //////////////
    // var new_icon = $('#new_icon').get(0); // this is for adding a button icon for full screen
    map1 = new google.maps.Map(document.getElementById('map_canvas1'),{
        // center: new google.maps.LatLng(39.9778, -86.2959),
        center: new google.maps.LatLng(39.9778, -86.44),
        zoom: 10.5,
        disableDefaultUI: true, //E:it disables all default icons from google map
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        mapTypeControl: false, //E: It disables type of map option
        mapTypeControlOptions: {
            style: google.maps.MapTypeControlStyle.HORIZONTAL_BAR
        //     position: google.maps.ControlPosition.TOP_CENTER
        },
        zoomControl: false,
        zoomControlOptions: {
            position: google.maps.ControlPosition.TOP_CENTER
        },
        // streetViewControl: false,
        // streetViewControlOptions: { position: google.maps.ControlPosition.LEFT_TOP},
        fullscreenControl: true,
        fullscreenControl: {
            position: google.maps.ControlPosition.BOTTOM_RIGHT
        },
```

```
81          scaleControl: false
82      });
83      // ////////////////////////// END Drawing MAIN MAP //////////////////
84
85      // ----------------- Start NEW Legend into the Main map---------------- //
86      // E: This part sets up a custom button into the map to display the legend
87      // These code-lines calls the js script located around 2620 in g2.php
88
89      var buttonOptions = {
90          gmap: map1,
91          name: 'Legend ',
92          position: google.maps.ControlPosition.TOP_RIGHT,
93          // action: function(){ //do something
94          //     // map1.panTo(new google.maps.LatLng(-33.9, 151.2));
95          //     jQuery('.feat_content').toggle('show');
96          //     report('m-clk*** ' , 'Main-map Legend ');
97          //     // jQuery(alert("button added.. "));
98          //     // alert("button added.. ");
99          // },
100     };
101     var button1 = new buttonControl(buttonOptions);
102
103     //// This is a second button. By clicking on it you get a new position view of the map
104     // var buttonOptions = {
105     //     gmap: map1,
106     //     name: 'Home',
107     //     position: google.maps.ControlPosition.TOP_RIGHT,
108     //     action: function(){map1.panTo(new google.maps.LatLng(-33.9, 151.2));}
109     // };
110     // var button1 = new buttonControl(buttonOptions);
111
112     //  ---------------- end NEW Legend ----------------- //
113
114
115     // E: This three lines add new icon into map to replace the usual 'fullscreen' icon provided by google map
116     $('#fullscreen').click(function() {
117         $('#map_canvas1 div.gm-style button[title="Toggle fullscreen view"]').trigger('click');
118         //// Add fullscreen events. When the full-screen is closed, it is registered.(It does not work for Firefox)
119         // document.addEventListener('webkitfullscreenchange', exitHandler);
120         // document.addEventListener('mozfullscreenchange', exitHandler);
121         // function exitHandler() {
122         //     if (!document.webkitIsFullScreen && !document.mozFullScreen) {
123         //         report('m-clk*** ' , 'out');
124         //         // document.removeEventListener("webkitfullscreenchange", exitHandler);
125         //     }
126         // }
127
128         // Add fullscreen's events. When they are out (close fullscreen), they are captured and reported into the DB
129         if (document.addEventListener) {
130             document.addEventListener('webkitfullscreenchange', exitHandler, false);
131             document.addEventListener('mozfullscreenchange', exitHandler, false);
132             document.addEventListener('fullscreenchange', exitHandler, false);
133             document.addEventListener('MSFullscreenChange', exitHandler, false);
134         }
135         // This function captures the close-fullscreen and reports into the DDBB
136         function exitHandler() {
137             if (!document.webkitIsFullScreen && !document.mozFullScreen && !document.msFullscreenElement) {
138                 report('m-clk*** ','close_fullscreen');
139                 document.removeEventListener("webkitfullscreenchange", exitHandler);
140                 document.removeEventListener('mozfullscreenchange', exitHandler);
141                 document.removeEventListener('fullscreenchange', exitHandler);
142                 document.removeEventListener('MSFullscreenChange', exitHandler);
143             }
144         }
145
146     });
147
148
149     // // ======= Start Add NEW button into the main map (NOT IN USE FOR NOW) ========//
150     // var BMP_buttonOptions = {
151     //     gmap: map1,
152     //     name: 'Legend+ ',
153     //     // position: google.maps.ControlPosition.TOP_RIGHT,
154     //     position: google.maps.ControlPosition.TOP_LEFT,
155     //     action: function(){
156     //         // map1.panTo(new google.maps.LatLng(-33.9, 151.2));
157     //         // jQuery('.feat_content').toggle('show');
158     //         alert("button added.. ");
159     //     }
160     // };
161     // var BMP_buttons = new BMP_buttonControl(BMP_buttonOptions);
```

```
162       // // =========  End Add NEW button into the main map ======= //
163
164
165       // $$$$$$$$$$$$    create the check box items for the MAIN map $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
166
167       //create the check box items (1),(2),(3),(4),(5),(6),(7)
168       var frame = document.createElement('DIV');
169       frame.className = 'BMP_checkboxes_frame';
170       var container1 = document.createElement('DIV');
171       container1.className = 'BMP_checkboxes_container1';
172       container1.style.display = 'inline-flex';
173       var container2 = document.createElement('DIV');
174       container2.className = 'BMP_checkboxes_container2';
175       container2.style.display = 'inline-flex';
176
177       // ---  (1) Menu for "Crop Rotation (cr)"
178       var cr_checkOptions = { //cr: "Crop Rotation"
179           title: "On/Off Crop Rotation", //"This allows for multiple selection/toggling on/off",
180           id1: "CropRot", // For container.id
181           id2: "CropRotation", // For bDiv.id
182           label: "Crop Rotation  ", // "   text"
183           action: function(){
184               report('m-clk*c ' , 'Crop Rotation click ');
185               // alert('you clicked check Crop Rotation');
186           }
187       };
188       var cr_checkbox = new checkBox_CropRotation(cr_checkOptions); //cr: "Crop Rotation"
189       // ---  (1) End Menu for "Crop Rotation (cr)"
190
191       // ---  (2) Menu for "Cover Crop (cc)"
192       var cc_checkOptions = { //cc: "Cover Crop"
193           title: "On/Off Cover Crop", //"This allows for multiple selection/toggling on/off",
194           id1: "CoverCrop1",
195           id2: "CoverCrop2",
196           label: "Cover Crop  ", // "   text"
197           action: function(){
198               report('m-clk*c ' , 'Cover Crop click ');
199               // alert('you clicked check Cover Crop');
200           }
201       };
202       var cc_checkbox = new checkBox_CoverCrop(cc_checkOptions); //cc: "Cover Crop"
203       // ---  (2) End Menu for "Cover Crop (cc)"
204
205       // ---  (3) Menu for "Strip Cropping (sc)"
206       var sc_checkOptions = { //sc: "Strip Cropping"
207           title: "On/Off Strip Cropping",
208           id1: "StripCropping1",
209           id2: "StripCropping2",
210           label: "Strip Cropping  ", // "   text"
211           action: function(){
212               report('m-clk*c ' , 'Strip Cropping click ');
213               // alert('you clicked check Cover Crop');
214           }
215       };
216       var sc_checkbox = new checkBox_StripCropping(sc_checkOptions); //sc: "Strip Cropping"
217       // ---  (3) End Menu for "Strip Cropping (sc)"
218
219       // ---  (4) Menu for "Filter Strip (fs)"
220       var fs_checkOptions = { //fs: "filter strip"
221           title: "On/Off Filter Strip", //"This allows for multiple selection/toggling on/off",
222           id1: "FilterStr",
223           id2: "Filter",
224           label: "Filter Strips  ", // "   text"
225           action: function(){
226               report('m-clk*c ' , 'Filter Cropping click ');
227               // alert('you clicked check 1');
228           }
229       };
230       var fs_checkbox = new checkBox_FilterStrip(fs_checkOptions); //fs: "filter strip"
231       // ---  (4) End Menu for "Filter Strip (fs)"
232
233       // ---  (5) Menu for "Grass waterways (gw)"
234       var gw_checkOptions = { //gw: "Grasswaterways"
235           title: "On/Off Grass waterways", //"This allows for multiple selection/toggling on/off",
236           id1: "Grasswaterways1",
237           id2: "Grasswaterways2",
238           label: "Grasswaterways  ", // "   text"
239           action: function(){
240               report('m-clk*c ' , 'Grass Waterways click ');
241               // alert('you clicked check 1');
242           }
```

```
243       };
244       var gw_checkbox = new checkBox_Grasswaterways(gw_checkOptions); //gw: "Grass waterways"
245       // ---  (5) End Menu for "Grass waterways (gw)"
246
247       // ---  (6) Menu for "No Tillage (nt)"
248       var nt_checkOptions = { //nt: "No Tillage"
249           title: "On/Off No Tillage", //"This allows for multiple selection/toggling on/off",
250           id1: "NoTillage1",
251           id2: "NoTillage2",
252           label: "Conservation Tillage  ", // "   text"
253           action: function(){
254               report('m-clk*c ' , 'Conservation Tillage click ');
255               // alert('you clicked check 1');
256           }
257       };
258       var nt_checkbox = new checkBox_NoTillage(nt_checkOptions); //nt: "NoTillage"
259       // ---  (6) End Menu for "No Tillage (nt)"
260
261       // ---  (7) Menu for "Wetlands (wt)"
262       var wt_checkOptions = { //wt: "Wetlands"
263           title: "On/Off Wetlands", //"This allows for multiple selection/toggling on/off",
264           id1: "Wetlands1",
265           id2: "Wetlands2",
266           label: "Wetlands  ", // "   text"
267           action: function(){
268               report('m-clk*c ' , 'Wetlands click ');
269               // alert('you clicked check 1');
270           }
271       };
272       var wt_checkbox = new checkBox_Wetlands(wt_checkOptions); //wt: "Wetlands"
273       // ---  (7) End Menu for "Wetlands (wt)"
274
275       container2.appendChild(cr_checkbox); //E: (1) Append "Crop Rotation"
276       container2.appendChild(cc_checkbox); //E: (2) Append "Cover Crop"
277       container2.appendChild(sc_checkbox); //E: (3) Append "Strip Cropping"
278       container2.appendChild(fs_checkbox); //E: (4) Append "Filter Strip"
279       container2.appendChild(gw_checkbox); //E: (5) Append "Grasswaterways"
280       container2.appendChild(nt_checkbox); //E: (6) Append "No Tillage"
281       container2.appendChild(wt_checkbox); //E: (7) Append "Wetlands"
282
283       container1.appendChild(container2);
284       frame.appendChild(container1);
285
286       map1.controls[google.maps.ControlPosition.TOP_RIGHT].push(frame); //E: position of "frame" div into te map
287
288       google.maps.event.addDomListener(frame,'click',function(){ //E: Add event "frame" to the map
289       });
290
291       // // ****************  Start a Second way to get check button in the main-map *************
292       // // It calls the 'bmp1_ckbox_function' function at g2.php. Line: 2893
293       // var box_bmp1 = document.createElement('DIV');
294       // var box_bmp1_exit = new bmp1_ckbox_function(box_bmp1,map1);
295       // map1.controls[google.maps.ControlPosition.TOP_CENTER].push(box_bmp1);
296       // // ***********************  End Second way *****************
297
298       // =========  End create the check box items $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ ======= //
299
300
301       // map2 = new google.maps.Map(document.getElementById('map_canvas2'), {
302       //     center: new google.maps.LatLng(39.9778, -86.2959),
303       //     zoom: 11,
304       //     mapTypeId: google.maps.MapTypeId.ROADMAP
305       // });
306
307       ////////////////TEST/////////////////////////////////////
308       /*var numberone = new google.maps.LatLng(39.9778,-86.2959);
309       number = new google.maps.GroundOverlay('one.jpg', map1);
310       number.setMap(map1);*/
311       ////////////////////////////////////////////////////////////
312
313
314       google.maps.event.addListener(map1, 'click', function goToTimeMap1() {
315           // alert ("Suggestion " + (+oneMap + +1) + " - Outside watershed+"); // newalert
316           report('m-clk+', 'Sug:' + (+oneMap + +1) + '  Outside-watershed',';'); // track the suggestion and outside
317       });
318
319
320       ////////////   MAIN PART OF THIS CODE  ///////////////////////
321       //E: Call 'Background' function to draw the base map (subbasins map)
322       doBackground();
323
```

```
324       //I check through an array and if the title appears in it that means we have data for that BMP. Each function
325       // that is called makes the two maps. You will see them inside of each function. Only background doesnt do this because
    the background is the same with either map.
326       $.each(forMapArray, function(index, value) {
327           //alert("YES");
328           if (forMapArray[index]["Title"] == "crop_rotation") {
329               doCropRotation();
330           }
331       });
332
333       $.each(forMapArray, function(index, value) {
334           if (forMapArray[index]["Title"] == "cover_crops") {
335               //console.log(value);
336               doCoverCrops();
337           }
338       });
339
340       $.each(forMapArray, function(index, value) {
341           if (forMapArray[index]["Title"] == "strip_cropping") {
342               // alert("strip");
343               doStripCropping();
344           }
345       });
346
347       $.each(forMapArray, function(index, value) {
348           if (forMapArray[index]["Title"] == "filter_strips") {
349               doFilterStrips();
350           }
351       });
352
353       $.each(forMapArray, function(index, value) {
354           if (forMapArray[index]["Title"] == "grassed_waterway") {
355               doGrassWaterway();
356           }
357       });
358
359       $.each(forMapArray, function(index, value) {
360           if (forMapArray[index]["Title"] == "conservation_tillage") {
361               doConserveTillage();
362           }
363       });
364
365       $.each(forMapArray, function(index, value) {
366           if (forMapArray[index]["Title"] == "variable_area_wetlands") {
367               //alert("IN");
368               dobinaryWetlands();
369           }
370       });
371
372
373       // Next lines shows FUNCTIONS for drawing POLYGONS (sub-basins) and LINES & MARKERS (filterStrip, Wets, etc.)
374       //  (0) Base Map (Backgournd) (bk)
375       //  (1) Crop-rotation         (cr)
376       //  (2) Cover-crops           (cc)
377       //  (3) Strip-Cropping        (sc)
378       //  (4) Filter-strip          (fs)
379       //  (5) Grass waterways       (gw)
380       //  (6) No-Till (Till conservation)(nt)
381       //  (7) Wetlands              (wt)
382
383       ////////////////////// (0)  Begin BACKGROUND  ////////////////////////////////
384       // Lay background map
385
386       // function doBackground() {
387       //     doBack();
388       // }
389
390       // ----------- start 'doBack'
391       // function doBack() {
392       function doBackground() {
393           // Initialize JSONP request
394           var script = document.createElement('script');
395           var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
396           url.push('sql=');
397           //Streams
398           // var query = 'SELECT GRIDCODE, Area_Acres,Length_mil FROM ' +
399             //  '1pU7pdW8h9zLV6VUSdsrmmX47zAvF6BPVjYiShGA';
400           var query = 'SELECT GRIDCODE, geometry, Area_Acres,Length_mil FROM ' +
401               '1pU7pdW8h9zLV6VUSdsrmmX47zAvF6BPVjYiShGA';
402           var encodedQuery = encodeURIComponent(query);
403           url.push(encodedQuery);
```

```
404           url.push('&callback=drawBack'); //E:Here, the function 'drawBack' is called.  <===
405           url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
406           // url.push('&key=IzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
407           script.src = url.join('');
408           var body = document.getElementsByTagName('body')[0];
409           body.appendChild(script);
410       }
411       // ----------- end of 'doBack'
412
413       // ----------- start 'drawBack'
414       drawBack = function(data) {
415           var rows = data['rows'];
416           // alert("L-407 rows (Back): " + rows);//E:It show the list of sub-basins
417           // alert("rows: " + typeof 'rows');//alert("row: " + rows.length);//E: rows.length = 130
418           // alert("L-407 rows (Back): " + rows[0]);//E:alert. 'row[0]' is string and shows the sub-basin 129
419           // alert("Type row[0]: " + typeof 'rows[0]');//E:alert
420           // alert("row[0][1]['geometry']: " + rows[0][1]['geometry']);//E:alert
421
422               //var whichNode=100;
423               for (var i in rows) {
424                   var newCoordinates = [];
425                   var whichNode = "";
426                   //var geometries = rows[i][1]['geometry'];
427                   // alert (geometries);
428
429                   //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
430                   var newCoordinates = constructNewCoordinates(rows[i][1]['geometry']);
431                   //answersArray[oneMap].RATING
432                   var row = rows[i];
433                   var whichNode = row[0];
434                   // alert (whichNode);
435                   /////////You will put your acreage here//////////
436                   var acres = parseFloat(row[2]).toFixed(1);
437                   var rivers = parseFloat(row[3]).toFixed(1);
438                   /////////You will put your stream miles here////////
439                   //alert (whichNode);
440
441                   var background = new google.maps.Polygon({
442                       path: newCoordinates,
443                       //strokeColor: colors[0],
444                       strokeOpacity: .6,
445                       strokeWeight: 1,
446                       fillOpacity: 0,
447                       fillColor: "#ffffff",
448                       // fillColor: "#a182f1",//"#ffffff",
449                       clickable: true,
450                       indexID: whichNode
451                   });
452
453                   background.setMap(map1);
454                   //var listAll
455                   var obj = find(subBasinArray, 'subbasinID', whichNode);
456                   if (obj) {
457                       var listAll = "Sub-basin Area: " + acres + " acres | Stream Length+: " + rivers + " miles <br" +
458                           " />" + JSON.stringify(obj);
459                       listAll = listAll.replace(/"0.0"/g, "No");
460                       listAll = listAll.replace(/"1.0"/g, "Yes");
461                       listAll = listAll.replace(/,/g, "<br />");
462                       listAll = listAll.replace(/"/g, "");
463                       listAll = listAll.replace(/}/g, "");
464                       listAll = listAll.replace(/{/g, "");
465                       listAll = listAll.replace(/_/g, " ");
466                       listAll = listAll.replace(/variable area wetlands/g, "wetlands area+");
467                       listAll = listAll.replace(/:/g, ": ");
468                       listAll = listAll.replace(/variable wetfr wetlands/g, "wetlands drainage");
469                       listAll = listAll.replace(/wetlands area/g, "Wetlands area");
470                       listAll = listAll.replace(/filter strips/g, "Filter strip width in feet+");
471                       listAll = listAll.replace(/wetlands drainage/g, "Wetlands drainage area fraction+");
472                       listAll = listAll.replace(/subbasinID/g, "Sub-basin ID+");
473                       listAll = listAll.replace(obj.variable_area_wetlands, obj.variable_area_wetlands + " acres ");
474                       //alert (listAll);
475                   } else {
476                       var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + "Sub-
    basin ID: " + whichNode;
477                   }
478
479                   var obj = {
480                       'list': listAll
481                   };
482                   background.objInfo = obj;
483                   google.maps.event.addListener(background, 'click', function(event) {
```

```
                    console.log("in map 1");

                    $('.displayStuff').html(this.objInfo.list);
                    var abc = this.objInfo.list + '<br><div class="displayStuffb" name="What Do They Mean">' +
                        '<a target="_blank" href="infoBox.html" rel="shadowbox;height=640;width=620" name="What Do They Mean
">' +
                        '<strong><em name="What Do They Mean">What do these numbers mean?</em></strong></a></div>';
                    infowindow2 = new google.maps.InfoWindow({
                        content: abc,
                        position: event.latLng
                    });
                    infowindow2.open(map1);
                    /*setTimeout(function() {
                        infowindow2.close();
                    }, 5000);*/
                    // alert ("Suggestion: " + this.indexID); // newalert
                    report('m-clk+', 'Sug:' + (+oneMap + +1) + '  Sub-basin:' + this.indexID + ';'); // trackable

                    /*$.ajax({url: 'sendToTime.php',type: 'post',
                        data: "JSONHolder=" + "Map1" + "," + page + "," + session + "," + this.indexID,
                        success: function(data) {
                        }
                    }); //*/
                });

                backArray.push(background);
            }
            //map.fitBounds(bounds);
        };
        // ---------- end 'drawBack'
    ////////////////////////  EndBACKGROUND////////////////////////


    ///////////////// (1)  Begin Crop Rotation/////////////////////////////////////

    // function doCropRotation() {
    //     docrop1();
    // }
    // ------------  end 'doCropRotation'

    // ------------  start 'docrop1'
    // function docrop1() {
    function doCropRotation() {
        var obj = find(forMapArray, 'Title', 'crop_rotation');
        if (obj) {
            //alert("in it");
            var listofSubs = obj.subs;
            //var listofSubs = obj.subs;
            var strLen = listofSubs.length;
            var listofSubs = listofSubs.slice(0, strLen - 1);
            //alert(listofSubs);
        }

        // listOfSubs=forMapArray1["crop_rotation"];
        //alert(listOfSubs);
        // Initialize JSONP request
        var script = document.createElement('script');
        var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
        url.push('sql=');
        //Streams
        var query = 'SELECT GRIDCODE, geometry, Area_Acres,Length_mil FROM ' +
            '1pU7pdW8h9zLV6VUSdsrmmX47zAvF6BPVjYiShGA Where GRIDCODE in (' + listofSubs + ')';
        var encodedQuery = encodeURIComponent(query);
        url.push(encodedQuery);
        url.push('&callback=drawCrop1');//E:Here, the function 'drawCrop1' is called.  <===
        url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
        // url.push('&key=1ZBdUCcDAjB0w94aiSRybOvnbtUxHYrvaMmeljHaD');
        // 1ZBdUCcDAjB0w94aiSRybOvnbtUxHYrvaMmeljHaD
        // https://fusiontables.google.com/DataSource?docid=1ZBdUCcDAjB0w94aiSRybOvnbtUxHYrvaMmeljHaD#map:id=3
        script.src = url.join('');
        var body = document.getElementsByTagName('body')[0];
        body.appendChild(script);
    }
    // ------------  end 'docrop1'

    // ------------  start 'drawCrop1'

    drawCrop1 = function(data) {
        //function drawWet1(data) {
        //alert("ON");
        var rows = data['rows'];
```

File - C:\wamp64\www\wrestoreBCK2\model\js\mapping_new.js

```
564          //var whichNode=100;
565          for (var i in rows) {
566              var newCoordinates = [];
567              var whichNode = "";
568              //var geometries = rows[i][1]['geometry'];
569              //alert (geometries)
570
571              //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
572              var newCoordinates = constructNewCoordinates(rows[i][1]['geometry']);
573              //answersArray[oneMap].RATING
574              var row = rows[i];
575              var whichNode = row[0];
576              ////////You will put your acreage here//////////
577              var acres = parseFloat(row[2]).toFixed(1);
578              var rivers = parseFloat(row[3]).toFixed(1);
579              ////////You will put your stream miles here////////
580              //alert (whichNode);
581
582              crop = new google.maps.Polygon({
583                  path: newCoordinates,
584                  //strokeColor: colors[0],
585                  strokeOpacity: .4,
586                  strokeWeight: 1,
587                  fillOpacity: 1,
588                  fillColor: "#8da1bf",
589                  indexID: whichNode
590              });
591              crop.setMap(map1);
592
593              var obj = find(subBasinArray, 'subbasinID', whichNode);
594              if (obj) {
595                  var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + JSON.
    stringify(obj);
596                  listAll = listAll.replace(/"0.0"/g, "No");
597                  listAll = listAll.replace(/"1.0"/g, "Yes");
598                  listAll = listAll.replace(/,/g, "<br />");
599                  listAll = listAll.replace(/"/g, "");
600                  listAll = listAll.replace(/}/g, "");
601                  listAll = listAll.replace(/{/g, "");
602                  listAll = listAll.replace(/_/g, " ");
603                  listAll = listAll.replace(/variable area wetlands/g, "wetlands area");
604                  listAll = listAll.replace(/:/g, ": ");
605                  listAll = listAll.replace(/variable wetfr wetlands/g, "wetlands drainage");
606                  listAll = listAll.replace(/wetlands area/g, "Wetlands area");
607                  listAll = listAll.replace(/filter strips/g, "Filter strip width in feet");
608                  listAll = listAll.replace(/wetlands drainage/g, "Wetlands drainage area fraction");
609                  listAll = listAll.replace(/subbasinID/g, "Sub-basin ID");
610                  listAll = listAll.replace(obj.variable_area_wetlands, obj.variable_area_wetlands + " acres");
611                  //alert (listAll);
612              } else {
613                  var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + "Sub-basin
     ID: " + whichNode;
614              }
615
616              var obj = {
617                  'list': listAll
618              };
619              crop.objInfo = obj;
620              google.maps.event.addListener(crop, 'click', function(event) {
621                  //console.log(this.objInfo);
622                  $('.displayStuff').html(this.objInfo.list);
623                  var abc = this.objInfo.list + '<br><div class="displayStuffb" name="What Do They Mean"><a target="_blank"
    href="infoBox.html" rel="shadowbox;height=640;width=620" name="What Do They Mean"><strong><em name="What Do They Mean">What do
    these numbers mean?</em></strong></a></div>';
624                  infowindow2 = new google.maps.InfoWindow({
625                      content: abc,
626                      position: event.latLng
627                  });
628                  infowindow2.open(map1);
629                  /*setTimeout(function() {
630                          infowindow2.close();
631                      }, 5000);*/
632                  // alert (this.indexID); // newalert
633                  report('m-clk+', 'Sug:' + (+oneMap + +1) + '  Sub-basin:'+this.indexID+';'); // trackable
634
635                  /*$.ajax({url: 'sendToTime.php', type: 'post',
636                      data: "JSONHolder=" + "Map1" + "," + page + "," + session + "," + this.indexID,
637                      success: function(data) {
638                      }
639                  }); //*/
640
```

```
            });

            cropArray.push(crop);
        }
        //map.fitBounds(bounds);
    };

    // ------------  end 'drawCrop1'

    ////////////////////////////////End Crop Rotation totally ////////////////////////



    ////////////////////(2) Begin COVER CROPS  ////////////////////////////

    // function doCoverCrops() {
    //     docover1();
    // }
    // ------------  end 'doCoverCrops'

    // ------------  start 'docover1'
    // function docover1() {
    function doCoverCrops() {
        var obj = find(forMapArray, 'Title', 'cover_crops');
        if (obj) {
            //alert("in it");
            var listofSubs = obj.subs;
            //var listofSubs = obj.subs;
            var strLen = listofSubs.length;
            var listofSubs = listofSubs.slice(0, strLen - 1);
        }
        // Initialize JSONP request
        var script = document.createElement('script');
        var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
        url.push('sql=');
        //Streams
        var query = 'SELECT GRIDCODE, geometry, Area_Acres,Length_mil FROM ' +
            '1pU7pdW8h9zLV6VUSdsrmmX47zAvF6BPVjYiShGA Where GRIDCODE in (' + listofSubs + ')';
        var encodedQuery = encodeURIComponent(query);
        url.push(encodedQuery);
        url.push('&callback=drawCover1');//E:Here, the function 'drawCover1' is called.  <===
        url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
        script.src = url.join('');
        var body = document.getElementsByTagName('body')[0];
        body.appendChild(script);
    }
    // ------------  end 'docover1'

    // ------------  start 'drawCover1'

    drawCover1 = function(data) {
        //function drawWet1(data) {
        //alert("ON");
        var rows = data['rows'];
        //var whichNode=100;
        for (var i in rows) {
            var newCoordinates = [];
            var whichNode = "";
            //var geometries = rows[i][1]['geometry'];
            //alert (geometries)

            //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
            var newCoordinates = constructNewCoordinates(rows[i][1]['geometry']);
            //answersArray[oneMap].RATING
            var row = rows[i];
            var whichNode = row[0];
            ////////You will put your acreage here//////////
            var acres = parseFloat(row[2]).toFixed(1);
            var rivers = parseFloat(row[3]).toFixed(1);
            ////////You will put your stream miles here////////
            //alert (whichNode);

            cover = new google.maps.Polygon({
                path: newCoordinates,
                //strokeColor: colors[0],
                strokeOpacity: .4,
                strokeWeight: 1,
                fillOpacity: 1,
                fillColor: "#99c9ba",
                indexID: whichNode
            });
```

```
722             cover.setMap(map1);
723
724
725             var obj = find(subBasinArray, 'subbasinID', whichNode);
726             if (obj) {
727                 var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + JSON.
    stringify(obj);
728                 listAll = listAll.replace(/"0.0"/g, "No");
729                 listAll = listAll.replace(/"1.0"/g, "Yes");
730                 listAll = listAll.replace(/,/g, "<br />");
731                 listAll = listAll.replace(/"/g, "");
732                 listAll = listAll.replace(/}/g, "");
733                 listAll = listAll.replace(/{/g, "");
734                 listAll = listAll.replace(/_/g, " ");
735                 listAll = listAll.replace(/variable area wetlands/g, "wetlands area");
736                 listAll = listAll.replace(/:/g, ": ");
737                 listAll = listAll.replace(/variable wetfr wetlands/g, "wetlands drainage");
738                 listAll = listAll.replace(/wetlands area/g, "Wetlands area");
739                 listAll = listAll.replace(/filter strips/g, "Filter strip width in feet");
740                 listAll = listAll.replace(/wetlands drainage/g, "Wetlands drainage area fraction");
741                 listAll = listAll.replace(/subbasinID/g, "Sub-basin ID");
742                 listAll = listAll.replace(obj.variable_area_wetlands, obj.variable_area_wetlands + " acres");
743                 //alert (listAll);
744             } else {
745                 var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + "Sub-basin
     ID: " + whichNode;
746             }
747
748             var obj = {
749                 'list': listAll
750             };
751             cover.objInfo = obj;
752             google.maps.event.addListener(cover, 'click', function(event) {
753                 //console.log(this.objInfo);
754                 $('.displayStuff').html(this.objInfo.list);
755                 var abc = this.objInfo.list + '<br><div class="displayStuffb" name="What Do They Mean"><a target="_blank"
    href="infoBox.html" rel="shadowbox;height=640;width=620" name="What Do They Mean"><strong><em name="What Do They Mean">What do
    these numbers mean?</em></strong></a></div>';
756                 infowindow2 = new google.maps.InfoWindow({
757                     content: abc,
758                     position: event.latLng
759                 });
760                 infowindow2.open(map1);
761                 /*setTimeout(function() {
762                         infowindow2.close();
763                     }, 5000);*/
764                 // alert (this.indexID); // newalert
765                 report('m-clk+', 'Sug:' + (+oneMap + +1) + '  Sub-basin:'+this.indexID+';'); // trackable
766
767                 /*$.ajax({url: 'sendToTime.php',type: 'post',
768                     data: "JSONHolder=" + "Map1" + "," + page + "," + session + "," + this.indexID,
769                     success: function(data) {
770                     }
771                 });//*/
772             });
773
774             coverArray.push(cover);
775
776         }
777         //map.fitBounds(bounds);
778     };
779     // ------------  end 'drawCover1'
780
781     //////////////////////////  End COVER CROPS totally ////////////////////////
782
783
784     ////////////////////////// (3) Begin STRIP CROPPING ////////////////////////////////
785
786     // function doStripCropping() {
787     //     dostrip1();
788     // }
789     // ----------------  end 'doStripCropping()'
790
791     // ----------------  start 'dostrip1()'
792     // function dostrip1() {
793     function doStripCropping() {
794         var obj = find(forMapArray, 'Title', 'strip_cropping');
795         if (obj) {
796             //alert("in it");
797             var listofSubs = obj.subs;
798             //var listofSubs = obj.subs;
```

```
799             var strLen = listofSubs.length;
800             var listofSubs = listofSubs.slice(0, strLen - 1);
801         }
802         // Initialize JSONP request
803         var script = document.createElement('script');
804         var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
805         url.push('sql=');
806         //Streams
807         var query = 'SELECT GRIDCODE, geometry, Area_Acres,Length_mil FROM ' +
808             '1pU7pdW8h9zLV6VUSdsrmmX47zAvF6BPVjYiShGA Where GRIDCODE in (' + listofSubs + ')';
809         var encodedQuery = encodeURIComponent(query);
810         url.push(encodedQuery);
811         url.push('&callback=drawStrip1');//E:Here, the function 'drawStrip1' is called.  <===
812         url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
813         script.src = url.join('');
814         // alert("script.src: " + url);
815         var body = document.getElementsByTagName('body')[0];
816         body.appendChild(script);
817     }
818     // ----------------  end 'dostrip1()'
819
820     // ----------------  start 'drawStrip1' ---------------------- /GREEN/
821     drawStrip1 = function(data) {
822         //function drawWet1(data) {
823         //alert("ON");
824         var rows = data['rows'];
825         //var whichNode=100;
826         for (var i in rows) {
827             var newCoordinates = [];
828             var whichNode = "";
829             //var geometries = rows[i][1]['geometry'];
830             //alert (geometries)
831
832             //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
833             var newCoordinates = constructNewCoordinates(rows[i][1]['geometry']);
834             //answersArray[oneMap].RATING
835             var row = rows[i];
836             var whichNode = row[0];
837             /////////You will put your acreage here//////////
838             var acres = parseFloat(row[2]).toFixed(1);
839             var rivers = parseFloat(row[3]).toFixed(1);
840             /////////You will put your stream miles here////////
841             //alert (whichNode);
842
843             strip = new google.maps.Polygon({
844                 path: newCoordinates,
845                 //strokeColor: colors[0],
846                 strokeOpacity: .4,
847                 strokeWeight: 1,
848                 fillOpacity: 1,
849                 fillColor: "#87b07e",
850                 indexID: whichNode
851             });
852             strip.setMap(map1);
853             var obj = find(subBasinArray, 'subbasinID', whichNode);
854             var jonArray = [];
855             if (obj) {
856                 var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + JSON.
    stringify(obj);
857                 listAll = listAll.replace(/"0.0"/g, "No");
858                 listAll = listAll.replace(/"1.0"/g, "Yes");
859                 listAll = listAll.replace(/,/g, "<br />");
860                 listAll = listAll.replace(/"/g, "");
861                 listAll = listAll.replace(/}/g, "");
862                 listAll = listAll.replace(/{/g, "");
863                 listAll = listAll.replace(/_/g, " ");
864                 listAll = listAll.replace(/variable area wetlands/g, "wetlands area");
865                 listAll = listAll.replace(/:/g, ": ");
866                 listAll = listAll.replace(/variable wetfr wetlands/g, "wetlands drainage");
867                 listAll = listAll.replace(/wetlands area/g, "Wetlands area");
868                 listAll = listAll.replace(/filter strips/g, "Filter strip width in feet");
869                 listAll = listAll.replace(/wetlands drainage/g, "Wetlands drainage area fraction");
870                 listAll = listAll.replace(/subbasinID/g, "Sub-basin ID");
871                 listAll = listAll.replace(obj.variable_area_wetlands, obj.variable_area_wetlands + " acres");
872                 //alert (listAll);
873             } else {
874                 var listAll = "Sub-basin Area: " + acres + " acres | Stream Length: " + rivers + " miles <br />" + "Sub-basin
     ID: " + whichNode;
875             }
876
877             var obj = {
```

```
878                 'list': listAll
879             };
880             strip.objInfo = obj;
881             google.maps.event.addListener(strip, 'click', function(event) {
882                 //console.log(this.objInfo);
883                 $('.displayStuff').html(this.objInfo.list);
884                 var abc = this.objInfo.list + '<br><div class="displayStuffb" name="What Do They Mean"><a target="_blank" href="infoBox.html" rel="shadowbox;height=640;width=620" name="What Do They Mean"><strong><em name="What Do They Mean">What do these numbers mean?</em></strong></a></div>';
885                 infowindow2 = new google.maps.InfoWindow({
886                     content: abc,
887                     position: event.latLng
888 
889                 });
890                 infowindow2.open(map1);
891                 /*setTimeout(function() {
892                         infowindow2.close();
893                     }, 5000);*/
894                 // alert (this.indexID); // newalert
895                 report('m-clk+', 'Sug:' + (+oneMap + +1) + '  Sub-basin:'+this.indexID+';'); // trackable
896 
897                 /*$.ajax({url: 'sendToTime.php', type: 'post',
898                     data: "JSONHolder=" + "Map1" + "," + page + "," + session + "," + this.indexID,
899                     success: function(data) {
900                     }
901                 }); //*/
902             });
903 
904 
905             stripArray.push(strip);
906         }
907         //map.fitBounds(bounds);
908     };
909     // ---------------  end 'drawStrip1'
910 
911     ////////////////////// End STRIP CROPPING totally ////////////////////////////////
912 
913 
914     ////////////////// (4) Begin FILTER STRIP///THIS ONE NEEDS TO BE THE POLYLINES//////////////////////////
915     // function doFilterStrips() {
916     //     dofilter1();
917     // }
918 
919     // -----------  start 'dofilter1'
920     // function dofilter1() {
921     function doFilterStrips() {
922 
923         var obj = find(forMapArray, 'Title', 'filter_strips');
924         if (obj) {
925             //alert("in it");
926             var listofSubs = obj.subs;
927             //var listofSubs = obj.subs;
928             var strLen = listofSubs.length;
929             var listofSubs = listofSubs.slice(0, strLen - 1);
930         }
931         // Initialize JSONP request
932         var script = document.createElement('script');
933         var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
934         url.push('sql=');
935         //Streams
936         var query = 'SELECT GRID_CODE, geometry FROM ' +
937             '15rPfCYXIoquDLpunT66cvEz2yaqw7R6BRKptqBQ Where GRID_CODE in (' + listofSubs + ')';
938         var encodedQuery = encodeURIComponent(query);
939         url.push(encodedQuery);
940         url.push('&callback=drawFilter1'); //E:Here, the function 'drawFilter1' is called.  <===
941         url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
942         script.src = url.join('');
943         var body = document.getElementsByTagName('body')[0];
944         body.appendChild(script);
945     }
946     // -----------  end 'dofilter1'
947 
948     // -----------  start 'drawfilter1'
949 
950     drawFilter1 = function(data) {
951         //function drawWet1(data) {
952         //alert("ON");
953         var rows = data['rows'];
954         //var whichNode=100;
955         //alert(JSON.stringify(subBasinArray));
956         for (var i in rows) {
```

```
            var newCoordinates = [];
            var whichNode = "";
            var row = rows[i];
            var whichNode = row[0];

            var obj = find(subBasinArray, 'subbasinID', whichNode);
            if (obj) {
                filterAcre = obj.filter_strips;
                //alert(filterAcre);

                switch (true) {
                    case (filterAcre == 0):
                        filterColor = "";
                        break;

                    case ((filterAcre > 0) && (filterAcre < 3)):
                        filterColor = "#e927c2";
                        break;

                    case ((filterAcre >= 3) && (filterAcre < 6)):
                        filterColor = "#bf8811";
                        break;

                    case ((filterAcre >= 6) && (filterAcre < 10)):
                        filterColor = "#7da569";
                        break;

                    case ((filterAcre >= 10) && (filterAcre < 13)):
                        filterColor = "#602288";
                        break;

                    case (filterAcre >= 13):
                        filterColor = "#b10c0c";
                        break;
                    default:
                        filterColor = "";
                        break;
                }

                //alert(wetlandsIcon + ":" + wetlandsize);
            }

            //alert (whichNode);
            //var geometries = rows[i][1]['geometry'];
            //alert (geometries)

            //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
            var newCoordinates = constructNewCoordinates(rows[i][1]['geometry']);
            //answersArray[oneMap].RATING

            filter = new google.maps.Polyline({
                path: newCoordinates,
                strokeColor: filterColor,
                strokeOpacity: 1,
                strokeWeight: 2,
                fillColor: "#daca43"
            });
            filter.setMap(map1);
            filterArray.push(filter);
        }
        //map.fitBounds(bounds);
    };

    // -----------  end 'drawfilter1'

    ////////////////////////////  End 'FILTER STRIP' totally ////////////////////////


    ///////////////This is used to parse out the long lats for all the polygons/////////////////////////

    function constructNewCoordinates(polygon) {
        var newCoordinates = [];
        var coordinates = null;
        if (polygon['coordinates'])
            coordinates = polygon['coordinates'];
        if (coordinates.length == 1) {
            coordinates = coordinates[0];
            // alert("length = 1");
        }
        // alert(coordinates);
```

```
1038            for (var i in coordinates) {
1039                newCoordinates.push(
1040                    new google.maps.LatLng(coordinates[i][1], coordinates[i][0]));
1041                //bounds.extend(newCoordinates[i]);
1042            }
1043            return newCoordinates;
1044        }
1045        ////////// ----- end 'constructNewCoordinates(polygon)'  ////////
1046
1047
1048        ///////////////// (5)  Begin GRASS-WATERWAYS Markers  ////////////////////////////////////////////
1049        // function doGrassWaterway() {
1050        //     dograss1();
1051        // }
1052        // ------ end 'doGrassWaterway()'
1053
1054        // ------ start 'dograss1()'
1055        // function dograss1() {
1056        function doGrassWaterway() {
1057            var obj = find(forMapArray, 'Title', 'grassed_waterway');
1058            if (obj) {
1059                //alert("in it");
1060                var listofSubs = obj.subs;
1061                //var listofSubs = obj.subs;
1062                var strLen = listofSubs.length;
1063                var listofSubs = listofSubs.slice(0, strLen - 1);
1064            }
1065            // Initialize JSONP request
1066            var script = document.createElement('script');
1067            var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
1068            url.push('sql=');
1069            //Streams
1070            var query = 'SELECT GRIDCODE, geometry FROM ' +
1071                '1iRLpYHfW4L9ncVMvhL5HD5Pwcuu63MVmRBWtn7Y Where GRIDCODE in (' + listofSubs + ')';
1072            var encodedQuery = encodeURIComponent(query);
1073            url.push(encodedQuery);
1074            url.push('&callback=drawGrass1');
1075            url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
1076            script.src = url.join('');
1077            var body = document.getElementsByTagName('body')[0];
1078            body.appendChild(script);
1079        }
1080        // ------------------------------- end 'dograss1()'
1081
1082        // --------------------------------- start 'drawGrass1'
1083        drawGrass1 = function(data) {
1084            var rows = data['rows'];
1085            //var whichNode=100;
1086            for (var i in rows) {
1087                var newCoordinates = [];
1088                var whichNode = "";
1089                //var geometries = rows[i][1]['geometry'];
1090                //alert (geometries)
1091
1092                //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
1093                var newCoordinates = constructNewCoordinatesGrass(rows[i][1]['geometry']);
1094                //answersArray[oneMap].RATING
1095                var row = rows[i];
1096                var whichNode = row[0];
1097                //alert (whichNode);
1098
1099                grass = geo;
1100                //var country = new google.maps.Marker({
1101                // position:new google.maps.LatLng(newCoordinates),
1102                //map:map1,
1103                //icon: customIcons[1],
1104                //})
1105                //alert (country);
1106                grass.setMap(map1);
1107                grassArray.push(grass);
1108            }
1109            //map.fitBounds(bounds);
1110        }; // --------------------------------- end 'drawGrass1'
1111
1112
1113        ////////////This is the new piece that takes the markers and not shapes for Grass Waterways/////////////
1114        function constructNewCoordinatesGrass(polygon) {
1115            var geoOptions = {
1116                strokeColor: colors[0],
1117                strokeOpacity: 0.8,
1118                strokeWeight: 1,
```

```
            fillColor: colors[0],
            fillOpacity: 0.3,
            icon: grassIcon
        };
        var opts = geoOptions;
        var newCoordinates = [];
        var coordinates = null;
        if (polygon['coordinates']) {
            var coordinates = polygon['coordinates'];
            var options = opts || {};
            options.position = new google.maps.LatLng(coordinates[1], coordinates[0]);
            geo = new google.maps.Marker(options);
            return geo;
        }
    }
    // ------ end 'constructNewCoordinatesGrass(polygon)'

    ////////END GW (GRASS-WATERWAYS) totally ////////////////////


    ////////////////////////  (6) Begin No-Till Markers ////////////////////////////
    // function doConserveTillage() {
    //     dotill1();
    // }
    // ------- end 'doConserveTillage()'

    // ------- start 'dotill1()'
    // function dotill1() {
    function doConserveTillage() {
        var obj = find(forMapArray, 'Title', 'conservation_tillage');
        if (obj) {
            //alert("in it");
            var listofSubs = obj.subs;
            //var listofSubs = obj.subs;
            var strLen = listofSubs.length;
            var listofSubs = listofSubs.slice(0, strLen - 1);
            //alert(listofSubs);
        }
        // Initialize JSONP request
        var script = document.createElement('script');
        var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
        url.push('sql=');
        //Streams
        var query = 'SELECT GRIDCODE, geometry FROM ' +
            '1iRLpYHfW4L9ncVMvhL5HD5Pwcuu63MVmRBWtn7Y Where GRIDCODE in (' + listofSubs + ')';
        var encodedQuery = encodeURIComponent(query);
        url.push(encodedQuery);
        url.push('&callback=drawTill1');
        url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
        script.src = url.join('');
        var body = document.getElementsByTagName('body')[0];
        body.appendChild(script);
    }
    // ------- end 'dotill1()'

    //   ------- start 'drawTill1'
    drawTill1 = function(data) {
        var rows = data['rows'];
        //var whichNode=100;
        for (var i in rows) {
            var newCoordinates = [];
            var whichNode = "";
            var row = rows[i];
            var whichNode = row[0];
            //var geometries = rows[i][1]['geometry'];
            //alert (geometries)

            //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
            var newCoordinates = constructNewCoordinatesTill(rows[i][1]['geometry']);
            //answersArray[oneMap].RATING

            notill = geo;
            notill.setMap(map1);
            conserveArray.push(notill);
            // alert("In");
        }
        //map.fitBounds(bounds);
    };
    //   ------- end 'drawTill1'

```

```
1200        ////////////This is the new piece that takes the markers and not shapes for No Till//////////////
1201        function constructNewCoordinatesTill(polygon) {
1202            var geoOptions = {
1203                strokeColor: colors[0],
1204                strokeOpacity: 0.8,
1205                strokeWeight: 1,
1206                fillColor: colors[0],
1207                fillOpacity: 0.3,
1208                icon: tillIcon
1209            };
1210            var opts = geoOptions;
1211            var newCoordinates = [];
1212            var coordinates = null;
1213            if (polygon['coordinates']) {
1214                var coordinates = polygon['coordinates'];
1215                var options = opts || {};
1216                options.position = new google.maps.LatLng(coordinates[1], coordinates[0]);
1217                geo = new google.maps.Marker(options);
1218                return geo;
1219            }
1220        }
1221        // ------ end 'constructNewCoordinatesTill(polygon)'
1222
1223        /////// end NO-TILLAGE totally ////////////////////
1224
1225
1226
1227        // ////////////////////  (7) Begin WETLANDS Markers ////////////////////////////
1228        // function dobinaryWetlands() {
1229        //     dowetlands1();
1230        // }
1231        // -------------- end 'dobinaryWetlands()'
1232
1233        // -------------- start 'dowetlands1()'
1234        // function dowetlands1() {
1235        function dobinaryWetlands() {
1236            var obj = find(forMapArray, 'Title', 'variable_area_wetlands');
1237            if (obj) {
1238                //alert("in it");
1239                var listofSubs = obj.subs;
1240                //var listofSubs = obj.subs;
1241                var strLen = listofSubs.length;
1242                var listofSubs = listofSubs.slice(0, strLen - 1);
1243            }
1244            // Initialize JSONP request
1245            var script = document.createElement('script');
1246            var url = ['https://www.googleapis.com/fusiontables/v1/query?'];
1247            url.push('sql=');
1248            //Streams
1249            var query = 'SELECT GRID_CODE, geometry FROM ' +
1250                '1h0Pw3awyHJC1Eal7QHyP_FQZ6r7PpC5aUU0ybD0 Where GRID_CODE in (' + listofSubs + ')';
1251            var encodedQuery = encodeURIComponent(query);
1252            url.push(encodedQuery);
1253            url.push('&callback=drawWet1');//E:Here, the function 'drawWet1' is called.  <===
1254            url.push('&key=AIzaSyAm9yWCV7JPCTHCJut8whOjARd7pwROFDQ');
1255            script.src = url.join('');
1256            var body = document.getElementsByTagName('body')[0];
1257            body.appendChild(script);
1258        }
1259        // -------------- end 'dowetlands1()'
1260
1261        // -------------- start 'drawWet1'
1262        drawWet1 = function(data) {
1263            var rows = data['rows'];
1264            //var whichNode=100;
1265
1266            // function to convert SVG To Image(SVG) {
1267            function svg_to_img(arg1){
1268                // var svg = document.getElementById('svg6');
1269                var svg = document.getElementById(arg1);
1270                var xml = new XMLSerializer().serializeToString(svg);
1271                var svg64 = btoa(xml); //for utf8: btoa(unescape(encodeURIComponent(xml)))
1272                var b64start = '';
1273                var image64 = b64start + svg64;
1274                return image64;
1275            }
1276
1277            for (var i in rows) {
1278                var newCoordinates = [];
1279                var whichNode = "";
1280                var row = rows[i];
```

```
            var whichNode = row[0];
            var obj = find(subBasinArray, 'subbasinID', whichNode);
            // alert(JSON.stringify(subBasinArray));
            if (obj) {
                wetlandsize = obj.variable_area_wetlands;
                //alert (wetlandsize);
                switch (true) {
                    case (wetlandsize == 0):
                        wetlandsIcon = "";
                        break;

                    case (wetlandsize < 2):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands1.png";
                        wetlandsIcon = svg_to_img('svg1');
                        break;

                    case ((wetlandsize >= 2) && (wetlandsize < 6)):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands2.png";
                        wetlandsIcon = svg_to_img('svg2');
                        break;

                    case ((wetlandsize >= 6) && (wetlandsize < 11)):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands3.png";
                        wetlandsIcon = svg_to_img('svg3');
                        break;

                    case ((wetlandsize >= 11) && (wetlandsize < 15)):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands4.png";
                        wetlandsIcon = svg_to_img('svg4');
                        break;

                    case ((wetlandsize >= 15) && (wetlandsize < 29)):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands5.png";
                        wetlandsIcon = svg_to_img('svg5');
                        break;

                    case ((wetlandsize >= 29) && (wetlandsize < 40)):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands6.png";
                        wetlandsIcon = svg_to_img('svg6');
                        break;

                    case ((wetlandsize >= 40)):
                        // wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands7.png";
                        wetlandsIcon = svg_to_img('svg7');
                        break;

                    default:
                        wetlandsIcon = "http://wrestore.iupui.edu/model/images/wetlands.png";
                        break;
                    //return wetlandsIcon;
                }
                //alert(wetlandsIcon + ":" + wetlandsize);
            }

            //var geometries = rows[i][1]['geometry'];
            //alert (geometries)

            //if (i==1) alert("geometry "+i+":"+rows[i][1]['geometry']['coordinates']);
            var newCoordinates = constructNewCoordinatesWet(rows[i][1]['geometry']);
            //answersArray[oneMap].RATING

            //alert (whichNode);
            wetlands = geo;
            //var country = new google.maps.Marker({
            // position:new google.maps.LatLng(newCoordinates),
            //map:map1,
            //icon: customIcons[1],
            //})
            //alert (country);
            wetlands.setMap(map1);
            wetArray.push(wetlands);
        }
        //map.fitBounds(bounds);
    };
    // -------------- end 'drawWet1'

    ///////// This is the new piece that takes the markers and not shapes //////////
    function constructNewCoordinatesWet(polygon) {

        var geoOptions = {
            strokeColor: colors[0],
```

```
            strokeOpacity: 0.8,
            strokeWeight: 1,
            fillColor: colors[0],
            fillOpacity: 0.3,
            icon: wetlandsIcon
        };
        //alert(wetlandsIcon);
        var opts = geoOptions;
        var newCoordinates = [];
        var coordinates = null;
        if (polygon['coordinates']) {
            var coordinates = polygon['coordinates'];
            var options = opts || {};
            options.position = new google.maps.LatLng(coordinates[1], coordinates[0]);
            geo = new google.maps.Marker(options);
            return geo;
        }
    }
    // ------- end 'constructNewCoordinatesWet(polygon)'

    // ////////////////////// End WETLANDS totally //////////////////////

} ///END MAPPING FUNCTIONs

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// function toggleLayerNew(whichArray, whichArray2, mapName) {
function toggleLayerNew(whichArray, mapName) {
    if (mapName.getMap()) {
        // alert (cropArray);
        $.each(whichArray, function(index, value) {
            // alert(value);
            value.setMap(null);
        });
        // $.each(whichArray2, function(index, value) {
        //     value.setMap(null);
        // });

    } else {
        $.each(whichArray, function(index, value) {
            // alert(value);
            value.setMap(map1);
        });
        // $.each(whichArray2, function(index, value) {
        //     value.setMap(map2);
        // });
    }
}


//alert(JSON.stringify(forMapArray));
//Going to call the list of subbasins we need for this spot
function find(arr, key, value) {
    for (var i = 0, l = arr.length; i < l; i++) {
        if (arr[i][key] === value) {
            return arr[i];
        }
    }
    // return {}; // if you would want it null-safe
}

// ====================  FUNCTION FOR TRACKING CHECKBOXS in LEGEND = (Added by E.Noa) ================== //
//  (1) Crop-rotation         (cr)
//  (2) Cover-crops           (cc)
//  (3) Strip-Cropping        (sc)
//  (4) Filter-strip          (fs)
//  (5) Grass waterways       (gw)
//  (6) No-Till (Till conservation)(nt)
//  (7) Wetlands              (wt)

function track_check_cropRotation(){ // (1) Crop-rotation (cr)
    var value_name = document.getElementsByClassName("cr")[0].getAttribute("value"); // Cover-Crop (cc)
    if ($('input.cr').is(':checked')) {
        report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
    }else{
        report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
    }
}


function track_check_coverCrop(){ // (2) Cover-crops (cc)
```

File - C:\wamp64\www\wrestoreBCK2\model\js\mapping_new.js

```
1443        var value_name = document.getElementsByClassName("cc")[0].getAttribute("value"); // Cover-Crop (cc)
1444        if ($('input.cc').is(':checked')) {
1445            report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
1446        }else{
1447            report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
1448        }
1449    }
1450
1451
1452    function track_check_stripCropping(){ // (3) Strip-Cropping (sc)
1453        var value_name = document.getElementsByClassName("sc")[0].getAttribute("value"); // Cover-Crop (cc)
1454        if ($('input.sc').is(':checked')) {
1455            report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
1456        }else{
1457            report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
1458        }
1459    }
1460
1461
1462    function track_check_filterStrip(){ //(4) Filter-strip (fs)
1463        var value_name = document.getElementsByClassName("fs")[0].getAttribute("value"); // Filter-strip (fs)
1464        if ($('input.fs').is(':checked')) {
1465            report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
1466        }else{
1467            report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
1468        }
1469    }
1470
1471
1472    function track_check_grassWaterway(){ // (5) Grass waterways (gw)
1473        var value_name = document.getElementsByClassName("gw")[0].getAttribute("value"); // Cover-Crop (cc)
1474        if ($('input.gw').is(':checked')) {
1475            report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
1476        }else{
1477            report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
1478        }
1479    }
1480
1481
1482    function track_check_noTill(){ // (6) No-Till (Till conservation)(nt)
1483        var value_name = document.getElementsByClassName("nt")[0].getAttribute("value"); // Cover-Crop (cc)
1484        if ($('input.nt').is(':checked')) {
1485            report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
1486        }else{
1487            report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
1488        }
1489    }
1490
1491
1492    function track_check_wetland(){ // (7) Wetlands (wt)
1493        var value_name = document.getElementsByClassName("wt")[0].getAttribute("value"); // Cover-Crop (cc)
1494        if ($('input.wt').is(':checked')) {
1495            report('m-clk+', 'Check of ' + value_name + ';'); // report('Check of', ' Filter-strips' + ';');
1496        }else{
1497            report('m-clk+', 'Un-check of ' + value_name + ';');// report('Un-check of', ' Filter-strips' + ';');
1498        }
1499    }
1500
1501 // ================== END ->  FUNCTION FOR TRACKING CHECKBOXS in LEGEND = (Added by E.Noa) ================== //
1502
1503
1504
```