// Schacht Aslani $.ajax({ url:“https://www.kimonolabs.com/api/4t8vjgt4?apikey=9adbc380301d558ba05315f79fff2e31&authorization=G3GwZYmY1egVWJOekuI1ec6HrdI8lOOm”, crossDomain: true, dataType: “jsonp”, success: function (response) { // these are stub numbers from a test API

var totalMiles = 0;  
 var totalTrips = 0;  
 var totalCommutes = 0;  
 var nameArray = [];  
 var milesArray = [];  
 var tripsArray = [];  
 var commutesArray = [];  
  
 $.each(response.results.collection1, function(i, item) {  
 totalMiles = totalMiles + Math.abs(parseFloat(item.miles));  
 totalMiles = +((totalMiles).toFixed(2));  
 totalCommutes = totalCommutes + Math.abs(parseFloat(item.commutes));  
 totalTrips = totalTrips + Math.abs(parseFloat(item.trips));  
 milesArray.push(parseFloat(item.miles));  
 nameArray.push(item.name.text);  
 tripsArray.push(parseFloat(item.trips));  
 commutesArray.push(parseFloat(item.commutes));  
 });  
  
 $( "#saaMiles" ).append(totalMiles);  
 $( "#saaCommutes" ).append(totalCommutes);  
 $( "#saaTrips" ).append(totalTrips);  
  
 renderLeaderBoardChart('#saaMilesLeaderboard', '#BB0000', 'Miles', nameArray, milesArray);  
 renderLeaderBoardChart('#saaCommutesLeaderboard', '#BB0000', 'Commutes', nameArray, commutesArray);  
 renderLeaderBoardChart('#saaTripsLeaderboard', '#BB0000', 'Trips', nameArray, tripsArray);  
  
 $( "#lastupdated" ).append(response.thisversionrun);  
  
},  
error: function (xhr, status) {  
  
}

});

$.ajax({ url:“https://www.kimonolabs.com/api/306jlvsc?apikey=9adbc380301d558ba05315f79fff2e31&authorization=G3GwZYmY1egVWJOekuI1ec6HrdI8lOOm”, crossDomain: true, dataType: “jsonp”, success: function (response) {

var totalMiles = 0;  
 var totalTrips = 0;  
 var totalCommutes = 0;  
 var nameArray = [];  
 var milesArray = [];  
 var tripsArray = [];  
 var commutesArray = [];  
  
 $.each(response.results.collection1, function(i, item) {  
 totalMiles = totalMiles + Math.abs(parseFloat(item.miles));  
 totalMiles = +((totalMiles).toFixed(2));  
 totalCommutes = totalCommutes + Math.abs(parseFloat(item.commutes));  
 totalTrips = totalTrips + Math.abs(parseFloat(item.trips));  
 milesArray.push(parseFloat(item.miles));  
 nameArray.push(item.name.text);  
 tripsArray.push(parseFloat(item.trips));  
 commutesArray.push(parseFloat(item.commutes));  
 });  
  
 $( "#paeMiles" ).append(totalMiles);  
 $( "#paeCommutes" ).append(totalCommutes);  
 $( "#paeTrips" ).append(totalTrips);  
  
 renderLeaderBoardChart('#paeMilesLeaderboard', '#337ab7', 'Miles', nameArray, milesArray );  
 renderLeaderBoardChart('#paeCommutesLeaderboard', '#337ab7', 'Commutes', nameArray, commutesArray );  
 renderLeaderBoardChart('#paeTripsLeaderboard', '#337ab7', 'Trips', nameArray, tripsArray );  
}

});

function renderLeaderBoardChart(className, color, name, riders, miles){ $(className).highcharts({ chart: { type: ‘bar’, animation: false }, legend: { enabled: false }, credits: { enabled: false }, title: { text: ’’ }, xAxis: { categories: riders }, yAxis: { title: { text: name } }, series: [{ name: name, data: miles, animation: false, color: color }] }); }

function renderAngularGraph(className, percentage) { var gaugeOptions = {

chart: {  
 type: 'solidgauge',  
 animation: false  
 },  
  
 title: '',  
  
 pane: {  
 center: ['50%', '70%'],  
 size: '100%',  
 startAngle: -90,  
 endAngle: 90,  
 background: {  
 backgroundColor: (Highcharts.theme && Highcharts.theme.background2) || '#EEE',  
 innerRadius: '60%',  
 outerRadius: '100%',  
 shape: 'arc'  
 }  
 },  
  
 tooltip: {  
 enabled: false  
 },  
  
 // the value axis  
 yAxis: {  
 stops: [  
 [0.1, '#DF5353'], // red  
 [0.5, '#DDDF0D'], // yellow  
 [0.9, '#55BF3B'] // green  
 ],  
 lineWidth: 0,  
 minorTickInterval: null,  
 tickPixelInterval: 400,  
 tickWidth: 0,  
 title: {  
 y: -70  
 },  
 labels: {  
 y: 16  
 }  
 },  
  
 plotOptions: {  
 solidgauge: {  
 dataLabels: {  
 y: 15,  
 borderWidth: 0,  
 useHTML: true  
 }  
 }  
 }  
};  
  
// The speed gauge  
$(className).highcharts(Highcharts.merge(gaugeOptions, {  
 yAxis: {  
 min: 0,  
 max: 100,  
 title: {  
 text: ''  
 }  
 },  
  
 credits: {  
 enabled: false  
 },  
  
 series: [{  
 animation: false,  
 name: 'Participation %',  
 data: [percentage],  
 dataLabels: {  
 format: '<div style="text-align:center"><span style="font-size:25px;color:' +  
 ((Highcharts.theme && Highcharts.theme.contrastTextColor) || 'black') + '">{y}</span><br/>' +  
 '<span style="font-size:12px;color:silver">%</span></div>'  
 },  
 tooltip: {  
 valueSuffix: ' %'  
 }  
 }]  
  
}));

}

function getRandomInt(min, max) { return Math.floor(Math.random() \* (max - min)) + min; }

function possibleTrips() { var startDate = new Date(“2015-04-02”); var today = new Date(); var todayStatic = new Date(); console.log(today.toString()); var workingDays = workingDaysBetweenDates(startDate, today); var trips = workingDays \* 2; if (todayStatic.getHours() < 19){ trips–; } console.log(trips); return trips; }

function workingDaysBetweenDates(startDate, endDate) {

// Validate input  
if (endDate < startDate)  
 return 0;  
  
// Calculate days between dates  
var millisecondsPerDay = 86400 \* 1000; // Day in milliseconds  
startDate.setHours(0,0,0,1); // Start just after midnight  
endDate.setHours(23,59,59,999); // End just before midnight  
var diff = endDate - startDate; // Milliseconds between datetime objects   
var days = Math.ceil(diff / millisecondsPerDay);  
  
// Subtract two weekend days for every week in between  
var weeks = Math.floor(days / 7);  
var days = days - (weeks \* 2);  
  
// Handle special cases  
var startDay = startDate.getDay();  
var endDay = endDate.getDay();  
  
// Remove weekend not previously removed.   
if (startDay - endDay > 1)   
 days = days - 2;   
  
// Remove start day if span starts on Sunday but ends before Saturday  
if (startDay == 0 && endDay != 6)  
 days = days - 1   
   
// Remove end day if span ends on Saturday but starts after Sunday  
if (endDay == 6 && startDay != 0)  
 days = days - 1   
  
return days;

}