JavaScript Files: * aggregation.js * ajax.js * boundaries.js * chart.js * envirocar-analyser.js * filter.js * interactivity.js * map.js * smartinfowindow.js * table.js

aggregation.js

Functions	Description
`startAggregation()`	creates a json from the current measurements selection and sends this json to the `aggregation.php` file on the server, which starts the aggregation
`displayAggregationResults()`	uses the json returned from `aggregation.php` to call up `createAggregationTable(json)` and `createChartFromAggregation(json)`

ajax.js

Functions	Description
`getTrackInformation()`	checks if there are any more tracks to get the measurements for, calls up `sendTrackRequest()` if this is the case, stops the animation of the loading screen if this is not the case
`getXMLHttpRequest()`	returns a XMLHttpObject, considering the browser used
`processTrackInformation()`	waits for the respond from the server and sorts the measurements requested to the measurements array
`sendRequest()`	sends a request to the server with a json file used to perform the aggregation
`sendTrackRequest(url, param, trackId)`	sends a request to the server with the id of the track requested

boundaries.js

Variables	Description
`nrwBoundaries`	The boundaries of NRW in an array
`nrwPolygon`	Uses the `nrwBoundaries` to create a `google.maps.Polygon`
`everythingElse`	Array which contains the coordinates from -90,-90 to 90,90 as a rectangle

chart.js

Class LineChart

Constructor	Description
`LineChart()`	Create a Line Chart for time related data

Functions	Description
`addPoint(series, point)`	Add a Point to an existing series
`addSeries(title, visible, id, data)`	adds a new line to the chart, visible determines whether the line should be displayed right from the start, data has to fit the current data scheme
`getAllSeries()`	returns all the series of the chart in an array
`getChartOptions()`	returns the current Chart Options as JSON
`getSeries(name)`	returns the Series with the name given
`highlight(id)`	selects points of the given id
`initChart()`	renders the Chart to the chart div
`redraw()`	redraws the chart – only necessary if data has been added
`remove()`	removes the chart and destroys chart object
`setAxisCategories(axis, categories)`	set the categories of the axis('x' or 'y'), categories have to passed as string array
`setAxisTitle(axis, title)`	set the title of the axis('x' or 'y')
`setSubtitle(subtitle)`	set the subtitle of the Diagram
`setTitle(title)`	set the Title of the Diagram
`unselect()`	unselects all selected points
Getter and setter functions are not listed here.	

Class BarChart

Constructor	Description
`BarChart()`	Create a Bar Chart

Functions	Description
`getAllSeries()`	returns all the series of the chart in an array
`getChartOptions()`	returns the current Chart Options as JSON
`getSeries(name)`	returns the Series with the name given
`initChart()`	renders the Chart to the chart div
`redraw()`	redraws the chart – only necessary if data has been added
`remove()`	removes the chart and destroys chart object
`setAxisCategories(axis, categories)`	set the categories of the axis('x' or 'y'), categories have to passed as string array
`setAxisTitle(axis, title)`	set the title of the axis('x' or 'y')
`setSubtitle(subtitle)`	set the subtitle of the Diagram
`setTitle(title)`	set the Title of the Diagram
Getter and setter functions are not listed here.	

---

envirocar-analyser.js

Classes: * `Phenomenon` * `Sensor` * `Measurement` * `Filter` * `Boundingbox` * `Query`

Class Phenomenon

Constructor	Description
`Phenomenon(name, unit, lowerLimit, upperLimit)`	Create a Phenomenon object with limits
`Phenomenon(name, unit)`	Create a Phenomenon object without limits

Variables	Type	Description
`name`	String	Name –> TODO: would not support multi-language. Maybe name_de and name_en instead?
`unit`	String	Unit (e.g. “km/h”)
`lowerLimit`	Number	Lower limit
`upperLimit`	Number	Upper limit

Functions	Description
`toString()`	Return the phenomenon as a String
`equals(otherPhenomenon:Phenomenon)`	Compare two phenomenons
Getter and setter functions are not listed here.	

Class Sensor

Constructor	Description
`Sensor(type, id, model, fuelType, manufacturer, constructionYear)`	Create a Sensor object

Variables	Type	Description
`id`	String	ID of the sensor
`model`	String	Car model
`fuelType`	String	Car’s fuel type (e.g. “gasoline”)
`manufacturer`	String	Car manufacturer
`constructionYear`	Number	Car’s construction year

Functions	Description
`parseJSON(json:JSON object)`	Parse a JSON object into a Sensor object
Getter and setter functions are not listed here.	

Class Measurement

Constructor	Description
`Measurement(id, point, timestamp, phenomenons, values)`	Create a Measurement object

Variables	Type	Description
`id`	String	ID of the measurement
`point`	`google.maps.Point`	Position
`timestamp`	Date	Date and time
`phenomenons`	Array	Array of used phenomenons
`values`	Array	Array of the values belonging to the `phenomenons`

Functions	Description
`toString()`	Return the measurement as a String
`equals(otherMeasurement:Measurement)`	Compare two measurements
`inLimitInterval()`	Return an array of the phenomenons which are in the interval given by the `lowerLimit` and `upperLimit`
`outOfLimitInterval()`	Return an array of the phenomenons which are out of the interval given by the `lowerLimit` and `upperLimit` In the second dimension of the array is a boolean value for: true: Value is higher than `upperLimit` false: Value is less than `lowerLimit`
Getter and setter functions are not listed here.	

Class Filter

Constructor	Description
	|

Variables	Type	Description

Functions	Description
`createUrlValue()`	Return the URL value with the given parameters inserted (e.g. “?bbox=…”)

Class Boundingbox

Constructor	Description
`Boundingbox(minX, minY, maxX, maxY)`	Create a Boundingbox object with two points
`Boundingbox(latLngBounds)`	Create a Boundingbox object with a `google.maps.LatLngBounds`

Variables	Type	Description
`minX`	Number	Minimum x-coordinate
`minY`	Number	Minimum y-coordinate
`maxX`	Number	Maximum x-coordinate
`maxY`	Number	Maximum y-coordinate

Functions	Description
`toUrlValue()`	Return the boundingbox as a String which is URL readable

Class Query

Constructor	Description
`Query(keyword)`	Create a Query object without a filter To see the possible keywords look at the function `getData()`
`Query(keyword, filter)`	Create a Query object with a filter To see the possible keywords look at the function `getData()`

Variables	Type	Description
`url`	String	URL to the enviroCar server **Must not be changed!**
`keyword`	String	To see the possible keywords look at the function `getData()`
`filter`	Filter	Filter to get only the preferred measurements etc.

Functions	Description
`getData()`	Get the data depending on the used `filter` and `keyword` Possible keywords: – measurements – sensors – tracks (not implemented yet)
`getMeasurements()`	Get measurements via the `/measurements` URL and return them as an array
`getSenors()`	Get sensors via the `/sensors` URL and return them as an array

---

filter.js

Variables	Type	Description
`startDate`	String	The start date specified by the user.
`endDate`	String	The end date specified by the user.
`baseUrl`	String	The base URL used by the temporal filter.
`envirocarTrackUrl`	String	The URL responsing all measured tracks from the envirocar API.
`baseUrlBBox`	String	The base URL used by the spatial filter.
`BBoxPrefix`	String	The bounding box prefix needed to build a URL for querying spatial-temporal data.
`rectangleActive`	Boolean	A boolean indicating if the user wants to use the spatial filter.

Functions	Description
`getDateTime()`	NOT IMPLEMENTED/OUTDATED – This function builds the URL string for the temporal filter, query the API, parses the data and display it in the trackSelectionList.
`getBBox()`	NOT IMPLEMENTED/OUTDATED – This function builds the URL string for the spatial filter, query the API, parses the data and display it in the trackSelectionList.
`setRectangleActive()`	Sets the boolean rectangleActive to true if the user wants to have a spatial filter.
`setRectangleNonActive()`	Sets the boolean rectangleActive to false if the user does not want to have a spatial filter.
`getDT()`	Building the URL string for the temporal filter.
`getBB()`	Building a string embodying the coordinates of the bounding box.
`getDateTimeBBox()`	Checks which type of filter the user wants to use, calling the appropriate method and creating the specific URL string.
`getLatestTracks()`	Getting the latest track from the envirocar API, parses it, looks up date and time, subtracting 24 hours from the time of the latest track and building a URL string in order to get the last measured 24 hours.

---

interactivity.js

Variables	Type	Description
`viewMode`	String	the view mode set to table
`streetmode`	Boolean	true, if the user enables the streetmode, otherwise false
`alerted`	Boolean	alert for the user concerning the use of street selection in OSM
`toggled`	Boolean	toggle for the content of the analyser-panel, standard state: false
`lastContent`	String	the last content in the analyser-panel, e.g. help, contact, imprint or terms of usage

Functions	Description
`changeMode()`	change the website from standard exploration mode to analysis mode
`datetimepicker()`	jQuery function to select a time range
`getWindowWidth()`	function to get the width of the browser window (body)
`getWindowHeight()`	function to get the height of the browser window (body)
`getScrollXY()`	function to get the scroll position of the page
`streetMode()`	enable or disable the street selection mode
`toggleAnalyserPanel(id)`	change the content of the analyser panel. Parameter id must specify the id of the element
`toggleHelp()`	enable or disable the help content in the analyser-panel
`toggleContact()`	enable or disable the contact content in the analyser-panel
`toggleImprint()`	enable or disable the imprint content in the analyser-panel
`toggleTerms()`	enable or disable the terms of usage content in the analyser-panel
`loadCarModels()`	load car models into Dual Listbox
`clearArray()`	clears the array and the HTML element where the cars are stored
`showProgressAnimation()`	show loading window
`hideProgressAnimation()`	hide loading window
`cancelEvent()`	cancel an onClick event, here: loading data
`popupwindow(url, title, w, h)`	create a popup window at the center of the screen
`limitFilter()`	open the popup window with the filter selection

map.js

Variables	Type	Description
`map`	`google.maps.Map`	The map on the website
`nrwBounds`	`google.maps.LatLngBounds`	Bounds that contain NRW
`markers`	array	Array of all markers displayed on the map
`mc`	`MarkerClusterer`	Clusters markers at specific zoom levels
`mcUsedBefore`	Boolean	Gives information whether the `mc` has been used at the last zoom level
`maxZoomLevelForClusterer`	Number	Stores the maximum zoom level for the `mc`
`markersBounds`	`google.maps.LatLngBounds`	Bounds that contain all displayed markers
`path`	`google.maps.MVCArray`	Stores the polyline underlying lat_lng values while street segment selection
`service`	`google.maps.DirectionsService`	Gives the polyline, “snapped to road”, between user selected points while selecting street segments
`poly`	`google.maps.Polyline`	Displays the polyline on the map based on `path`
`streetmode`	Boolean	Stores wether street segment selection is on or off
`alerted`	Boolean	Stores if the user was informed how to use the street segment selection
`streetlistener`	`google.maps.event.addListener`	Listener for adding the street segment selection
`removepointlistener`	`google.maps.event.addListener`	Listener for removing the user selected street segments by user
`polyexport`	`google.maps.MVCArray`	Stores the polyline made by user for export; identical to path except lifetime
`removepoints`	array	stores the length of `path`: Adds a history to the polyline and the selected points to enable rollback operations while selecting street segments
`carModelsExists`	Boolean	shows if a carModel object exists

Functions	Description

`applyAllFilter()`	checks which filter are active and removes all measurements which do not fit the filter settings, then calls up redrawData()
`applyCarSelection()`	removes all measurements from cars which are not selected by the car selection feature
`applyLimitFilter(phenomenon, min, max)`	takes over the settings from the limit filter popup and removes all measurements which do not fit the limit filter settings
`displayCoSpeedRatioMarkers()`	executes the enviro filter by evaluating all measurements by their speed/co2 ratio, the result is displayed on map, table and chart
`focusTrack()`	zooms to the track selected from the track selection list and removes all other measurements
`initMap()`	Initialize the map Must be called when the website is loaded
`limitFilterActive()`	returns true if there is a limit filter active, false if not
`resizeMap()`	Called when the map is resized Bounds will be changed to contain all markers in the viewport
`showMarkers(query:Query)`	Show measurements as markers on the map
`showMarkersClassified()`	displays the markers in red, green or yellow, considering the settings from the limit filter
`redrawData(marker, cars, chart, table, tracks)`	resets and redraws map, car selection, chart, table and tracks
`refreshMarkers(zoom:Number)`	Refresh the markers on the map depending on the current zoom level: Display markers on map for zoom levels higher than `maxZoomLevelForClusterer`, otherwise display them via the `mc`
`trackSelectionActive()`	returns true if a single track is selected from the track selection list, false if not
`buildInfoWindow(marker,map,measurements,val1,val2,val3,val4,phen1,phen2,phen3,phen4`	
`carSelectionActive()`	returns true if the car selection filter is used, returns false if no such filter is active
`checkCoSpeedValuesAvailability()`	enables or disables the enviroFilter function, considering the amount and uniformity of the speed and co2 values
	Removes all Marker and

`clearOverlays()`	Marker Clusterer from the map
`collectStreets(controlDiv, map:google.maps.Map)`	
`enableStreetmode()`	Enable the street segment selection: adds `removepointlistener`, `streetlistener`, sets `streetmode = true` and clears `polyexport`
`disableStreetmode()`	Disables the street segment selection: removes `removepointslistener`, `streetlistener`, sets `streetmode = false` and clears arrays except `polyexport`
`getPolylineAt(i:Number)`	Returns the Lat_Lng of `poly` at (i)
`getPolyline()`	Returns the polyline underlying google.maps.MVCArray
`interpolate()`	Starts the interpolation. Applies all filters. Checks wether trackselection is active, boundingbox is active or street selection was made
`interpolatePhen(idwkey)`	interpolates a single Phenomenon depending on the selection made for a specific phenomenon given by `idwkey`. Returns the marker array with the interpolated markers and corresponding infowindows
`showIdwSpeed()`	Displays the marker for the Speed interpolation
`showIdwCo2()`	Displays the marker for the Co2 interpolation
`showIdwConsumption()`	Displays the marker for the Consumption interpolation
`distance(p1,p2)`	Calculates distance between two points `p1`, `p2`. Returns the distance(`number`)
`classifyValues(measurements, idwkey)`	Classifies the values of the measurement array for the phenomenon given by `idwkey`. On basis of standard deviation function creates an array with the class breaks and returns it
`numSort(a,b)`	Helper function to sort numbers returns `(a-b)`
`buildSmallInfoWindow(idwmarker, map, interpolatedValues)`	Creates a new google.maps.InfoWindow for `idwmarker` with the interpolated Value given by `interpolatedValues`

smartinfowindow.js

Sourcecode from: https://code.google.com/p/gmaps-samples-v3/source/browse/trunk/smartinfowindow/?r=225 by pamela.fox added some changes concerning closure behaviour and offset values for displaying windows.

table.js

Functions	Description
`createTable()`	Create the table
`createAggregationTable(json)`	Create Table displaying only the aggregation results
`createCell(cell, text)`	Create DIV element and append to the table cell
`deleteTable()`	Delete the entire table out of the DOM
`deleteRows(value)`	Delete the rows of the table with a specific index
`initAggregationTable(json)`	Initialize special table for Aggregation Results
`initTable()`	Initialize the table
`openMarkerInfoWindow(id)`	Open the marker info window of the belonging measurement by triggering the click event
`ratiocolumn()`	Show the column with the co2 per kilometre values
`refreshTable()`	Refresh the table and display the chosen the phenomenons
`tablestyle()`	Use the existing HTML Table and changes some style parameters