Namespaces

RenderInfrastructure : `namespace`

Where the Rendering/Management related functions are

Querier : `namespace`

Where the querying related functions are

RenderInfrastructure : `namespace`

Where the Rendering/Management related functions are

Kind: global namespace

- RenderInfrastructure : `object`
  - .config(map, markerLayer, data, options)
  - .update()
  - .removeFeatureFromMap(featureId) ⇒ `boolean`
  - .addFeatureToMap(featureId) ⇒ `boolean`

RenderInfrastructure.config(map, markerLayer, data, options)

Sets up instance of renderer

Kind: static method of `RenderInfrastructure`

Param	Type	Description
map	`L.Map`	Leaflet map that will have things rendered to it
markerLayer	`L.markerClusterGroup`	Marker cluster that will contain markers
data	`JSON`	JSON that contains needed information for renderable things
options	`object`	object with attributes

RenderInfrastructure.update()

Call this when the map should be updated

Kind: static method of `RenderInfrastructure`

RenderInfrastructure.removeFeatureFromMap(featureId) ⇒ `boolean`

Removes a feature id from the map

Kind: static method of `RenderInfrastructure` **Returns**: `boolean` - true if feature was removed, false if not

Param	Type	Description
featureId	`string`	id which should be removed from map, ex: 'dam' or 'weir'

RenderInfrastructure.addFeatureToMap(featureId) ⇒ `boolean`

Adds a feature id to the map and forces an update

Kind: static method of `RenderInfrastructure` **Returns**: `boolean` - true if feature was added, false if JSON doesnt contain tag or objects is already being rendered

Param	Type	Description
featureId	`string`	id which should be added to map, ex: 'dam' or 'weir'

Querier : `namespace`

Where the querying related functions are

Kind: global namespace

- Querier : `object`
  - .queryGeoJsonFromServer(queryURL, bounds, isOsmData, callbackFn)
  - .createOverpassQueryURL(queryList, bounds, node_way_relation) ⇒ `string`

Querier.queryGeoJsonFromServer(queryURL, bounds, isOsmData, callbackFn)

Queries geoJSON or OSM Xml from an endpoint and returns it as geoJSON

Kind: static method of `Querier`

Param	Type	Description
queryURL	`string`	URL where geoJSON/Osm Xml is
bounds	`object`	(not necessary when using this function by itself) bounds object like: {north:?,east:?,south:?,west:?}
isOsmData	`boolean`	is the url going to return OSM Xml data? (such as overpass queries)
callbackFn	`function`	where the geoJSON will be sent on return, should be a 1-parameter function

Querier.createOverpassQueryURL(queryList, bounds, node_way_relation) ⇒ `string`

Creates a overpass query URL

Kind: static method of `Querier` **Returns**: `string` - a valid overpass URL

Param	Type	Description
queryList	`Array.<string>`	list of queries ex: ['waterway=dam','natural=lake']
bounds	`object`	bounds object in the form: {north:?,east:?,south:?,west:?}, which states WHERE to query
node_way_relation	`number`	binary choice for node,way,relation -- ex:111 = nodes, ways, AND relations -- 101 = nodes AND relations -- 100 = nodes only