```
/*
 * SADEK Serena
 * Juin 2017
 * TripTracker
 * templateMap.js
 *
 * Les fonctions de cette page concernent l'essenciel de ce qui concerne Google
 * Map API. Elle initialise la map et contient les fonctions faisant appel au
 * Serveur Google Map API.
 * Cette page contient également les fonctionalités développée lors de la préparation,
 * c'est à dire celle donc l'action est purement cosmétique
 */

var coll = true;
var creating = false;
var editing = null;

var creationMarkers = [];
var creationRoutes = [];

$(document).ready(function () {
    //Chargement de la carte
    initMap();

    //Tout les panneaux sont ouverts par défaut. Il faut donc les fermer manuellement
    $('#navDetails .slide-submenu').closest('.sidebar-body').hide();
    $('#navInsert .slide-submenu').closest('.sidebar-body').hide();
    $('#cmdInsert').show();

    //Lorsqu'on click sur une étape, le panneau de gauche doit se fermer et celui de
    droite s'ouvrir
    $('.list-group-item').click(function () {
        closeRight(true);
        $('#navDetails').show();
        openLeft();
    });

    //Lorsqu'on ouvre un panneau, on ferme l'autre

    $('.sidebar-left .slide-submenu').on('click', function () {
        closeRight(true);
    });

    $('.mini-submenu-left').on('click', function () {
        openRight();
        closeLeft(false);
    });

    $('#navDetails .slide-submenu').on('click', function () {
        closeLeft(true);
    });

    $('#cmdNavDetails').on('click', function () {
        openLeft();
        closeRight(true);
    });

    $('#cmdInsert').on('click', function () {
        closeLeft(false);
        closeRight(false);
        var height = $(window).height() - 50 - 37 - 7;
        $('#navInsert .insertPanel').css("max-height", height);
        window.setTimeout(function () {
            openAdd();
        }, 400);
    });

    $('#navInsert .slide-submenu').on('click', function () {
        closeAdd(true);
    });

```

```
71          $('body').on('show.bs.collapse', "#navTrips .collapse", function () {
72              if (coll) {
73                  coll = false;
74                  var id = "#" + (this.id);
75                  $('#navTrips .collapse').not(id).collapse("hide");
76                  coll = true;
77              }
78          });
79
80          //Lorsqu'on est en mode création, l'évènement click de la map déclenche une
81          //Recherche d'adresse
82          google.maps.event.addListener(map, 'click', function (event) {
83              if (focus !== null) { //Si l'utilisateur a son focus sur une étape
84                  getAdresseFromPosition(event.latLng);
85              }
86          });
87
88          /**
89           * Lorsque la position de la map change (que ce soit le zoom ou le pan), on
90           * vérifie si le limite de la map ne dépassent pas le lite du pôle nord et
91           * du pôle sud. Si c'est le cas, on décale le centre de manière à ce que la
92           * map soit de nouveau dans les limites.
93           */
94          google.maps.event.addListener(map, 'center_changed', function (event) {
95              var boundHeight = map.getBounds().f; // Limite haute et basse de la map
96              var initalCentre = map.getCenter();
97
98              if (boundHeight.b < -85) { //Limite basse de la map hors champ
99                  var ecart = -(boundHeight.b + 85);
100                 var centre = new google.maps.LatLng(
101                         initalCentre.lat() + ecart,
102                         initalCentre.lng()
103                         );
104                 map.setCenter(centre);
105             } else if (boundHeight.f > 85) { //Limite haute de la map hors champ
106                 var ecart = boundHeight.f - 85;
107                 var centre = new google.maps.LatLng(
108                         initalCentre.lat() - ecart,
109                         initalCentre.lng()
110                         );
111                 map.setCenter(centre);
112             }
113         });
114     });
115
116     /**
117      * Ouvre le panneau de droite
118      * @returns {undefined}
119      */
120     function openLeft() {
121         if ($('#navDetails .sidebar-body').is(":visible") == false) {
122             window.setTimeout(function () {
123                 var height = $(window).height() - 50 - 37 - 7;
124                 $('#navDetails .panel-body').css("max-height", height);
125
126                 $('#navDetails .sidebar-body').toggle();
127                 $('#cmdNavDetails').hide();
128             }, 450);
129         }
130     }
131
132     /**
133      * Ouvre le panneau de gauche
134      * @returns {undefined}
135      */
136     function openRight() {
137         if ($('.sidebar-left .sidebar-body').is(":visible") == false) {
138             if (ActivePanelId !== null && ActivePanelId < 5 && ActivePanelId >= 0) {
139                 panOnTrip(ActivePanelId);
140             }
141             window.setTimeout(function () {
```

```
142                 $('.sidebar-left .sidebar-body').toggle();
143                 $('.mini-submenu-left').hide();
144             }), 450;
145         }
146     }
147 
148     /**
149      * Ferme le panneau de gauche
150      * @param {type} reopen : La miniature doit-elle s'affiche pour permettre une
         réouverture ?
151      * @returns {undefined}
152      */
153     function closeRight(reopen) {
154         if ($('.sidebar-left .sidebar-body').is(":visible") == true) {
155             $('.sidebar-left .slide-submenu').closest('.sidebar-body').fadeOut('slide');
156             if (reopen) {
157                 $('.mini-submenu-left').fadeIn();
158             }
159         } else {
160             if (!reopen) {
161                 $('.mini-submenu-left').hide();
162             }
163         }
164 
165     }
166 
167     /**
168      * Ferme le panneau de droite
169      * @param {type} reopen : La miniature doit-elle s'affiche pour permettre une
         réouverture ?
170      * @returns {undefined}
171      */
172     function closeLeft(reopen) {
173         if ($('#navDetails .sidebar-body').is(":visible") == true) {
174             $('#navDetails .slide-submenu').closest('.sidebar-body').fadeOut('slide');
175             if (reopen) {
176                 $('#cmdNavDetails').fadeIn();
177             }
178         } else {
179             if (!reopen) {
180                 $('#cmdNavDetails').hide();
181             }
182         }
183     }
184 
185     /**
186      * Ouvre la panneau d'ajout
187      * @returns {undefined}
188      */
189     function openAdd() {
190         if ($('#navInsert .sidebar-body').is(":visible") == false) {
191             window.setTimeout(function () {
192                 $('#navInsert .sidebar-body').toggle();
193                 $('#cmdInsert').hide();
194             }), 400;
195         }
196         if (!creating) {
197             unsetPageDisplay();
198             creating = true;
199             ActivePanelId = null;
200             creationMarkers = [];
201             creationRoutes = [];
202             count = 0;
203         }
204     }
205     /**
206      * Ferme le panneau d'ajout, avec la possibilité ou non de l'ouvrir à nouveau
207      * @param {type} reopen
208      * @returns {undefined}
209      */
210     function closeAdd(reopen) {
```

```
        if ($('#navInsert .sidebar-body').is(":visible") == true) {
            $('#navInsert .slide-submenu').closest('.sidebar-body').fadeOut('slide');
            $('#cmdInsert').fadeIn();
        } else {
            if (!reopen) {
                $('#cmdInsert').hide();
                creating = false;
                creationMarkers = [];
                creationRoutes = [];
                count = 0;
                openRight();
            }
        }
    }

    /**
     * Inistalise la map principale
     * @returns {undefined}
     */
    function initMap() {
        var mapOptions = {
            zoom: 5,
            center: {lat: -34.397, lng: 150.644},
            mapTypeControl: false,
            streetViewControl: false,
            minZoom: 2,
            sensor: false,
            zoomControlOptions: {
                position: google.maps.ControlPosition.LEFT_BOTTOM
            },
            styles: [{"featureType": "landscape.natural", "elementType":
            "geometry.fill", "stylers": [{"visibility": "on"}, {"color": "#e0efef"}]},
            {"featureType": "poi", "elementType": "geometry.fill", "stylers":
            [{"visibility": "on"}, {"hue": "#1900ff"}, {"color": "#c0e8e8"}]},
            {"featureType": "road", "elementType": "geometry", "stylers": [{"lightness":
            100}, {"visibility": "simplified"}]}, {"featureType": "road", "elementType":
            "labels", "stylers": [{"visibility": "off"}]}, {"featureType":
            "transit.line", "elementType": "geometry", "stylers": [{"visibility": "on"},
            {"lightness": 700}]}, {"featureType": "water", "elementType": "all",
            "stylers": [{"color": "#7dcdcd"}]}]
        };
        var mapElement = document.getElementById('map');

        map = new google.maps.Map(mapElement, mapOptions);
    }


    /**
     * Récupère les informations complètes liée à la position et complète les
     * informations du panel sélectionné
     * @param {type} Position
     * @returns {undefined}
     */
    function getAdresseFromPosition(Position) {
        var geocoder = new google.maps.Geocoder();
        geocoder.geocode({'location': Position}, function (results, status) {
            if (status === 'OK') {
                if (results[1]) {
                    placeMarker(Position, results[0].formatted_address);
                    var id = "#adress" + focus;
                    $(id).val(results[1].formatted_address);
                } else {
                    alert('Pas d\'adresse réferencée pour cette position');
                }
            } else {
                window.alert('Geocoder failed due to: ' + status);
            }
        });
    }

    /**
```

```
273     * Récupère les informations complètes liée à l'adresse et complète les
274     * informations du panel sélectionné
275     * @param {type} Adresse
276     * @returns {undefined}
277     */
278    function getPositionFromAdresse(Adresse) {
279        var geocoder = new google.maps.Geocoder();
280        geocoder.geocode({'address': Adresse}, function (results, status) {
281            if (status === 'OK') {
282                var id = "#adress" + focus;
283                $(id).val(results[0].formatted_address);
284                placeMarker(new google.maps.LatLng(
285                        results[0].geometry.location.lat(),
286                        results[0].geometry.location.lng()),
287                        results[0].formatted_address);
288            } else {
289                window.alert('Geocoder failed due to: ' + status);
290            }
291        });
292    }
293
294    /**
295     * Géolocalise l'utilisateur, obtient les informations complète de cette
296     * localisation et complète les informations du panel sélectionné
297     * @returns {undefined}
298     */
299    function geoLocation() {
300        // Try HTML5 geolocation.
301        if (navigator.geolocation) {
302            navigator.geolocation.getCurrentPosition(function (position) {
303                var pos = new google.maps.LatLng(position.coords.latitude,
                    position.coords.longitude);
304                getAdresseFromPosition(pos);
305            }, function () {
306                alert("Une erreur est survenue lors de la géolocalisation");
307            });
308        } else {
309            alert("La géolocalisation n'est pas autorisée par votre navigateur.");
310        }
311    }
312
313    /**
314     * Place un marqueur sur la map en le stockang à la position correspondant au
315     * focus.
316     * @param {LatLng} location
317     * @param {string} address
318     * @returns {undefined}
319     */
320    function placeMarker(location, address) {
321        if (focus !== null) {
322            var id = null;
323            if (creationMarkers[focus] !== "none" && creationMarkers[focus] !== null) {
324                creationMarkers[focus].setMap(null);
325                if (typeof creationMarkers[focus].id !== "undefined") {
326                    id = creationMarkers[focus].id;
327                }
328            }
329            creationMarkers[focus] = new google.maps.Marker({
330                position: location,
331                map: map
332            });
333
334
335            if (id !== null) {
336                creationMarkers[focus].id = id;
337            }
338
339            creationMarkers[focus].address = address;
340
341            TraceRoute(focus);
342
```

```
343             PanOnCreationTrip();
344         }
345     }
346 
347     /**
348      * Tente de traces les deux routes auxquelles le marqueur pourrait être relié
349      * @param {int} PlaceId
350      * @returns {undefined}
351      */
352     function TraceRoute(PlaceId) {
353         TracePreviousRoad(PlaceId);
354         TraceNextRoad(PlaceId);
355     }
356 
357     /**
358      * Tente de tracer la route entre le marqueur et l'emplacement précédent
359      * @param {int} PlaceId
360      * @returns {undefined}
361      */
362     function TracePreviousRoad(PlaceId) {
363         if (PlaceId - 1 >= 0) {
364             var negaArray = creationMarkers.slice(0, PlaceId).reverse();
365             var count = PlaceId - 1;
366             negaArray.forEach(function (element) {
367                 if (creationMarkers[count] !== null) {
368                     if (creationMarkers[count] == "none") {
369                         return;
370                     } else {
371                         if (typeof creationRoutes[PlaceId - 1] == "object") {
372                             creationRoutes[PlaceId - 1].display.setMap(null);
373                         }
374                         setPath(creationMarkers[count], creationMarkers[PlaceId],
                            PlaceId - 1);
375                         if (true) {
376                             return;
377                         }
378                     }
379                 }
380                 count--;
381             });
382         }
383     }
384 
385     /**
386      * Tente de tracer la route entre le marqueur et l'emplacement suivant
387      * @param {int} PlaceId
388      * @returns {undefined}
389      */
390     function TraceNextRoad(PlaceId) {
391 
392         if (Number(PlaceId) + 1 < creationMarkers.length) {
393             var posiArray = creationMarkers.slice(Number(PlaceId) + 1,
            creationMarkers.length);
394             var count = Number(PlaceId) + 1;
395             posiArray.forEach(function (element) {
396                 if (creationMarkers[count] !== null) {
397                     if (creationMarkers[count] == "none") {
398                         return;
399                     } else {
400                         if (typeof creationRoutes[count - 1] == "object") {
401                             creationRoutes[count - 1].display.setMap(null);
402                         }
403                         setPath(creationMarkers[PlaceId], creationMarkers[count], count
                            - 1);
404                         if (true) {
405                             return;
406                         }
407                     }
408                 }
409                 count++;
410             });
```

```
411         }
412     }
413 
414     /**
415      * Prépare les paramètres de la fonction de géocodage, pour éviter
416      * que son caractère asyncrone lui fasse perdre des informaitons.
417      * @param {type} position1
418      * @param {type} position2
419      * @returns {undefined}
420      */
421     function setPath(position1, position2, StoragePosition) {
422         creationRoutes[StoragePosition] = [];
423         creationRoutes[StoragePosition].display = new
          google.maps.DirectionsRenderer({suppressMarkers: true, preserveViewport: true});
424 
425         var dep = new google.maps.LatLng(position1.position.lat(),
          position1.position.lng());
426         var arr = new google.maps.LatLng(position2.position.lat(),
          position2.position.lng());
427 
428         var directionsService = new google.maps.DirectionsService();
429 
430         creationRoutes[StoragePosition].display.setMap(map);
431 
432         var request = {
433             origin: dep,
434             destination: arr,
435             travelMode: google.maps.TravelMode.WALKING,
436             provideRouteAlternatives: false
437         };
438 
439 
440         directionsService.route(request, function (response, status) {
441             if (status === google.maps.DirectionsStatus.OK) {
442                 //Informations complètes concernant la route trouveée
443                 creationRoutes[StoragePosition].route = response;
444                 //Connecteur entre la réponse et la map
445                 creationRoutes[StoragePosition].display.setDirections(response);
446             }
447             if (status === "ZERO_RESULTS") {
448                 creationRoutes[StoragePosition].display.setMap(null);
449                 drawFlight(dep, arr, StoragePosition);
450             }
451         });
452     }
453 
454     /**
455      * Dessine une ligne droite entre deux point ne pouvant pas être reliés par la terre
456      * @returns {undefined}
457      */
458     function drawFlight(position1, position2, StoragePosition) {
459         creationRoutes[StoragePosition]
460         creationRoutes[StoragePosition].display = new google.maps.Polyline({
461             path: [position1, position2],
462             geodesic: true,
463             strokeColor: '#FF0000',
464             strokeOpacity: 1.0,
465             strokeWeight: 4
466         });
467         //creationRoutes[StoragePosition].display.setMap(map);
468         creationRoutes[StoragePosition].route = "polyline";
469         showFligh(StoragePosition);
470     }
471 
472     function showFligh(StoragePosition) {
473         creationRoutes[StoragePosition].display.setMap(map);
474     }
475 
476     /**
477      * Supprime les routes auxquel un point est lié
478      * @param {int} dotId
```

```
479     * @returns {undefined}
480     */
481    function suppressRoadsOfDot(dotId) {
482        var dot1 = null;
483
484        if (typeof creationRoutes[Number(dotId) - 1] !== "string") {
485            creationRoutes[Number(dotId) - 1].display.setMap(null);
486        }
487
488        var negaArray = creationMarkers.slice(0, Number(dotId)).reverse();
489        dot1 = Number(dotId) - 1;
490        var count = Number(dotId) - 1;
491        negaArray.forEach(function (element) {
492            if (creationMarkers[count] !== null) {
493                dot1 = count;
494                return;
495            }
496            count--;
497        });
498
499        if (true) {
500            creationRoutes[Number(dotId) - 1] = null;
501        }
502
503        if (Number(dotId) + 1 < creationMarkers.length) {
504            var posiArray = creationMarkers.slice(Number(dotId) + 1,
                creationMarkers.length);
505            var count = Number(dotId) + 1;
506            posiArray.forEach(function (element) {
507                if (creationMarkers[count] !== null) {
508                    if (creationMarkers[count] == "none") {
509                        return;
510                    } else {
511                        creationRoutes[count - 1].display.setMap(null);
512                        creationRoutes[count - 1] = null;
513                        if (dot1 !== null) {
514                            setPath(creationMarkers[dot1], creationMarkers[count], count
                                - 1);
515                        }
516                        if (true) {
517                            return;
518                        }
519                    }
520                }
521                count++;
522            });
523        }
524    }
525
526
527
```