Hướng dẫn tích hợp MapLibre trên nền bản đồ Goong trong web

**Tóm tắt**: Tài liệu dưới đây trình bày cách tính hợp MapLibretrên nền bản đồ của Goong, và sử dụng các dịch vụ cơ bản của Goong, bao gồm:

1. Hiện các kiểu bản đồ: cơ bản, vệ tinh, tối, sáng,... gắn marker, vẽ vòng tròn bao quanh marker.

2. Tìm kiếm: nhập tên địa chỉ, hiển thị các gợi ý liên quan tới tên địa chỉ nhập, sau khi chọn thì nhảy ra điểm đó trên bản đồ (sử dụng autocomplete tìm kiếm gợi ý, rồi dùng place-detail để lấy thông tin tọa độ về địa chỉ đó).

3. Dẫn đường: nhập tọa độ điểm đầu và cuối, hiển thị đường dẫn trên bản đó, có thông tin về khoảng cách và thời gian di chuyển (direction với phương tiên di chuyển: car, taxi,..).

**Cụ thể**:

1. **Các tham số cần thiết:**

Cần có: map key, api key

(<https://account.goong.io/keys>)

1. **Khởi tạo bản đồ và các chế độ hiện thị**

* Trong trang .html thêm 2 phần sau trong header để rend mapbox:

<link rel='stylesheet' href='https://unpkg.com/maplibre-gl/dist/maplibre-gl.css' />

<script src='https://unpkg.com/maplibre-gl/dist/maplibre-gl.js'></script>

* Khởi tạo bản đồ:

const map = new maplibregl.Map({

container: 'map',

// change style of map by change the link below of Goong map style, there are 5 styles.

style: `${mapUrl}goong\_map\_web.json?api\_key=${mapKey}`,

zoom: zoom,

center: center

});

* Chọn kiểu bản đồ: hàm changeStyle, truyền vào 1 trong 5 kiểu bản đồ

function changeStyle(styleUrl, name) {

map.setStyle(styleUrl);

document.querySelector('.popover-button').textContent = name;

togglePopover(); // Close the popover after selecting a style

}

**styleUrl** là link của từng loại bản đồ, ví dụ:

kiểu Normal có link: [https://tiles.goong.io/sources/goong.json?api\_key={your\_map\_key](https://tiles.goong.io/sources/goong.json?api_key={your_api_key)}, kiểu vệ tinh: <https://tiles.goong.io/assets/goong_satellite.json?api_key={your_map_key>},..

* Vẽ marker và đường tròn xung quanh điểm khởi tạo ban đầu

map.on('load', () => {

const circleData = {

'type': 'FeatureCollection',

'features': [{

'type': 'Feature',

'geometry': {

'type': 'Polygon',

'coordinates': [drawCircle(center, radiusInMeters)]

}

}]

};

map.addSource('circle', {

'type': 'geojson',

'data': circleData

});

new maplibregl.Marker()

.setLngLat(center)

.addTo(map);

map.addLayer({

id: 'circle',

type: 'fill',

source: 'circle',

layout: {},

paint: {

'fill-color': '#588888',

'fill-opacity': 0.5

}

});

});

* Trong hàm trên thực hiện 2 việc: gắn marker lên bản đồ, vẽ vòng tròn.

Khi gắn marker thì chỉ cần khởi tạo: new maplibregl.Marker(), và gắn tọa độ.

Khi vẽ đường tròn: bản chất là vẽ 1 lớp layer có vòng tròn được tô màu, rồi sử dụng addLayer để hiển thị nó lên bản đồ. Hàm vẽ vòng tròn drawCircle:

function drawCircle(center, radiusInMeters) {

const points = 64;

const coords = {

latitude: center[1],

longitude: center[0]

};

const km = radiusInMeters / 1000;

const ret = [];

const distanceX = km / (111.320 \* Math.cos(coords.latitude \* Math.PI / 180));

const distanceY = km / 110.574;

let theta, x, y;

for (let i = 0; i < points; i++) {

theta = (i / points) \* (2 \* Math.PI);

x = distanceX \* Math.cos(theta);

y = distanceY \* Math.sin(theta);

ret.push([coords.longitude + x, coords.latitude + y]);

}

ret.push(ret[0]);

return ret;

}

Khi vẽ vòng tròn, bản chất là vẽ các điểm rồi nối lại với nhau thành 1 đa giác, nên càng nhiều điểm vòng tròn sẽ càng mịn, ví dụ trên vẽ vòng tròn với points = 64 là đủ rồi.

Chú ý: cần truyền center có long trước lat và kiểu là số, ví dụ: [105.85242472181584, 21.029579719995272], khi đưa lên vẽ bản đồ, thì tọa độ điểm sẽ là long trước lat, còn khi gửi điểm vào link gọi các api của Goong sẽ là lat trước long.

1. **Tìm kiếm địa điểm.**

* Với bất kỳ tên địa chỉ nào người dùng nhập vào, hiển thị các gợi ý cho người dùng chọn. Hàm fetchDataAutoComplete sẽ gọi thực hiện điều đó.

function fetchDataAutoComplete(query) {

const apiLink = `${apiUrl}/Place/AutoComplete?api\_key=${apiKey}&input=${encodeURIComponent(query)}`;

fetch(apiLink)

.then(response => response.json())

.then(data => {

if (data.predictions) {

renderArray(data.predictions.map(prediction => prediction));

} else {

renderArray([]);

}

})

.catch(error => {

console.error('Error fetching data:', error);

renderArray([]); // Render an empty array in case of error

});

}

* Khi người dùng nhập tên địa chỉ, thì hàm trên sẽ gọi dịch vụ Autocomplete của Goong để trả về các gợi ý về tên địa chỉ ứng với từ mà người dùng nhập, kèm theo đó là place\_id. Sau đó hiển thị những gợi ý lên cho người dùng chọn, khi chọn thì gọi tiếp dịch vụ place detail bằng tham số place\_id, thì sẽ lấy được tọa độ của điểm này. Hàm fetchPlaceDetails:

function fetchPlaceDetails(placeId) {

const apiLink = `${apiUrl}/Place/Detail?api\_key=${apiKey}&place\_id=${placeId}`;

fetch(apiLink)

.then(response => response.json())

.then(data => {

if (data.result) {

const { location } = data.result.geometry;

const lngLat = [location.lng, location.lat];

addMarkerAndCircleAround(lngLat)

} else {

console.error('No result found for place details');

}

})

.catch(error => {

console.error('Error fetching place details:', error);

});

}

* Với tọa đồ của điểm mà người dùng đã chọn, ta sẽ gán marker, vẽ vòng tròn cho điểm đó y như lúc khởi tạo ban đầu và thêm 1 bước nữa là bản đồ phải bay ra điểm mới này nữa.

hàm addMarkerAndCircleAround:

function addMarkerAndCircleAround(lngLat) {

// Add marker

console.log(lngLat)

new maplibregl.Marker()

.setLngLat(lngLat)

.addTo(map);

// Draw circle

if (map.getLayer('circle')) {

map.removeLayer('circle');

map.removeSource('circle');

}

const circleData = {

'type': 'FeatureCollection',

'features': [{

'type': 'Feature',

'geometry': {

'type': 'Polygon',

'coordinates': [drawCircle(lngLat, radiusInMeters)]

}

}]

};

map.addSource('circle', {

'type': 'geojson',

'data': circleData

});

map.addLayer({

'id': 'circle',

'type': 'fill',

'source': 'circle',

'layout': {},

'paint': {

'fill-color': '#588888',

'fill-opacity': 0.5

}

});

// fly to this location

map.flyTo({ center: lngLat, zoom: zoom });

}

Lưu ý: Số lần gọi autocomplete thì cần phải tối ưu, tùy theo nhu cầu của ứng dụng, ví dụ theo kiểu sau 2,3 ký tự mới đc gọi hoặc khách nhập nhưng chỉ gọi sau 3s không nhập gì. Vì Goong sẽ tính phí mỗi lần gọi autocomplete này.

1. **Dẫn đường**

Dẫn đường sử dụng dịch vụ direction, gửi kèm tọa độ điểm đầu, điểm cuối, phương tiện di chuyển, nhận về mã đường đi, khoảng cách 2 điểm, thời gian đi dự kiến,...

* Sau khi khách hàng nhập tọa độ điểm đầu và cuối, gọi dịch vụ directiopn của bên Goong sau đó sẽ giải mã đường đi và hiển thị đường đi đó lên bản đồ.

Hàm fetchDirections gọi dịch vụ của Goong:

function fetchDirections(startCoords, endCoords) {

const apiLink = `${apiUrl}/Direction?origin=${startCoords}&destination=${endCoords}&vehicle=${directionVehicle}&api\_key=${apiKey}`;

fetch(apiLink)

.then(response => response.json())

.then(data => {

if (data.routes && data.routes.length > 0) {

const route = data.routes[0].overview\_polyline.points;

const distance = data.routes[0].legs[0].distance.text;

const time = data.routes[0].legs[0].duration.text;

const midLocation = data.routes[0].legs[0].steps[Math.floor(data.routes[0].legs[0].steps.length / 2)].end\_location //get a location in middle to display popup distance and time direction

const midCoords = [midLocation.lng, midLocation.lat]

const decodedRoute = decodePolyline(route);

displayRoute(decodedRoute, startCoords, endCoords, distance, time);

} else {

alert('Could not find a route.');

}

})

.catch(error => {

console.error('Error fetching directions:', error);

alert('Error fetching directions.');

});

}

Hàm này sẽ gọi dịch vụ direction của Goong, bằng cách truyền tham số tọa độ điểm đầu và cuối, phương tiên di chuyển (ở đây lấy car), và sẽ lấy ra được đường (route), khoảng cách 2 điểm (distance), thời gian đi dự kiến (time),...

* Sau đó cần giải mã đường đi (route), hàm decodePolyline:

function decodePolyline(encoded) {

var points = [];

var index = 0, len = encoded.length;

var lat = 0, lng = 0;

while (index < len) {

var b, shift = 0, result = 0;

do {

b = encoded.charAt(index++).charCodeAt(0) - 63;

result |= (b & 0x1f) << shift;

shift += 5;

} while (b >= 0x20);

var dlat = ((result & 1) ? ~(result >> 1) : (result >> 1));

lat += dlat;

shift = 0;

result = 0;

do {

b = encoded.charAt(index++).charCodeAt(0) - 63;

result |= (b & 0x1f) << shift;

shift += 5;

} while (b >= 0x20);

var dlng = ((result & 1) ? ~(result >> 1) : (result >> 1));

lng += dlng;

points.push([lng \* 1e-5, lat \* 1e-5]);

}

return points;

}

Giải mã ra được 1 mảng các tọa độ điểm mà đường sẽ đi qua.

* Hiển thị đường đó lên bản đồ, kèm thông tin về khoảng cách và thời gian đi dự kiến, hàm displayRoute

function displayRoute(route, startCoords, endCoords, distance, time) {

if (map.getSource('route')) {

map.removeLayer('route');

map.removeSource('route');

}

const start = startCoords.split(',').map(x => Number(x));

const end = endCoords.split(',').map(x => Number(x));

const longLatStart = [start[1], start[0]];

const longLatEnd = [end[1], end[0]];

//add marker to start and end location

addMarker(longLatStart)

addMarker(longLatEnd)

map.addSource('route', {

'type': 'geojson',

'data': {

'type': 'Feature',

'properties': {},

'geometry': {

'type': 'LineString',

'coordinates': route

}

}

});

map.addLayer({

'id': 'route',

'type': 'line',

'source': 'route',

'layout': {

'line-join': 'round',

'line-cap': 'round'

},

'paint': {

'line-color': '#3887be',

'line-width': 5,

'line-opacity': 0.9

}

});

// Find the midpoint of the route to show popup

const midPoint = route[Math.floor(route.length / 2)];

// Add a marker for the midpoint with distance + time information

new maplibregl.Popup()

.setLngLat(midPoint)

.setHTML(`<p>Khoảng cách: ${distance}</p> <p>Thời gian: ${time}</p> <br/>`)

.addTo(map);

map.fitBounds(route.reduce(function (bounds, coord) {

return bounds.extend(coord);

}, new maplibregl.LngLatBounds(route[0], route[0])));

}

Hàm trên làm 3 việc, gán marker vào điểm đầu và cuối, vẽ đường và vẽ popup hiển thị thông tin khoảng cách và thời gian.

Lưu ý: điểm khi gán trên bản đồ phải là long trước lat: [105.85242472181584, 21.029579719995272], Vị trí hiển thị popup đang để là giữa quãng đường, có thể thay đổi bằng cách lấy giá trị midPoint khác, nhưng nên đều lấy trong mảng route.