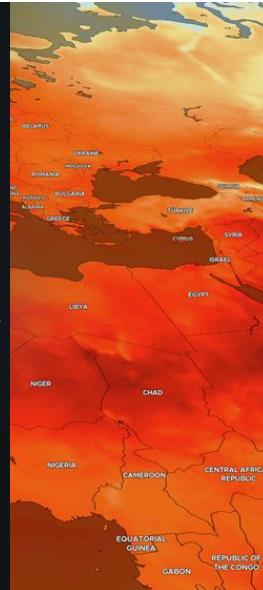


# MapTiler SDK JS, la experiencia MapLibre con esteroides



```
const map = new Map({  
  container,  
  style: MapStyle.OUTDOOR,  
  terrain: true,  
  terrainExaggeration: 1.5,  
  terrainControl: true,  
  geolocateControl: true,  
  geolocate: true,  
})
```



# MapLibre GL JS v4.x

## Features y mejoras

- Eliminar todos los getters y setters globales de `maplibregl`, los siguientes métodos han cambiado:
  - a. `maplibregl.version` => `getVersion()`
  - b. `maplibregl.workerCount` => `getWorkerCount()` , `setWorkerCount(...)`
  - c. `maplibregl.maxParallelImageRequests` => `getMaxParallelImageRequests()` , `setMaxParallelImageRequests (...)`
  - d. `maplibregl.workerUrl` => `getWorkerUrl()` , `setWorkerUrl(...)`
- La atribución está activada por defecto, `MapOptions.attributionControl` ahora es del tipo que maneja el control, se eliminó `MapOptions.customAttribution`
- Se cambió la interfaz de `Source` para devolver promesas en lugar de callbacks
- Se modificó el método `map.loadImage` para devolver una prome en lugar de callback
- Se agregó la opción `opacidad` y el método `setOpacity` a los marcadores
- Nuevo método `setUrl` en RasterTileSource para actualizar dinámicamente el recurso TileJSON existente
- Actualización de los métodos `addClass` y `removeClass` del `Popup` para devolver una instancia de Popup
- Se agregó el método `getData` para los Source GeoJSON para poder obtener todas las features del source
- Mejoras en el terreno y corrección de errores

# MapLibre GL JS

v3.x

```
map.loadImage
```

```
...  
  
map.on('load', function () {  
    map.loadImage('https://upload.wikimedia.org/wikipedia/commons/7/7c/201408_cat.png',  
        function (error, image) {  
            if (error) throw error;  
            map.addImage('cat', image);  
            ...  
            map.addLayer({  
                'id': 'points',  
                'type': 'symbol',  
                'source': 'point',  
                'layout': {  
                    'icon-image': 'cat',  
                    'icon-size': 0.25  
                }  
            });  
        }  
    );  
});
```

# MapLibre GL JS

v4.x

```
map.loadImage
```

```
map.on('load', async function () {
  const image = await map.loadImage(
    'https://upload.wikimedia.org/wikipedia/commons/7/7c/201408_cat.png');

  map.addImage('cat', image.data);
  ...

  map.addLayer({
    'id': 'points',
    'type': 'symbol',
    'source': 'point',
    'layout': {
      'icon-image': 'cat',
      'icon-size': 0.25
    }
  });
});
```

# MapLibre GL JS

## Globe



A screenshot of a globe visualization using MapLibre GL JS. The globe shows the world map with country boundaries and names. The Atlantic Ocean is visible at the top left. The map is styled with a light green for land and blue for water. A black sidebar on the right contains three colored dots (red, yellow, green) and a code snippet.

```
const map = new maplibregl.Map({
  container: 'map',
  projection: 'globe',
  zoom: 0,
  hash: true,
  center: [137.915089, 36.25956],
  style:
    'https://api.maptiler.com/maps/satellite/style.json?key=YOUR_MAPTILER_API_KEY_HERE',
});
```

# MapLibre GL JS

Sky (Atmosphere)



```
{  
    "version": 8,  
    "sprite": "https://api.maptiler.com/maps/satellite/sprite",  
    "glyphs": "https://api.maptiler.com/fonts/{fontstack}/{range}.pbf?key=YOUR_MAPTILER_API_KEY_HERE",  
    "metadata": {...},  
    "center": [...],  
    "zoom": 15,  
    "pitch": 85,  
    "sources": {...},  
    "layers": [...],  
    "terrain": {  
        "source": "terrain",  
        "exaggeration": 1  
    },  
    "sky": {  
        "sky-color": "#199EF3",  
        "fog-color": "#daeff0",  
        "horizon-blend": 1,  
        "fog-blend": 1  
    }  
}
```



# MapLibre GL JS

NEW

Documentation

The screenshot shows the MapLibre GL JS documentation homepage. At the top, there's a navigation bar with links for Overview, Introduction, Plugins, Style Specifications, API, Examples, and Guides. On the right side of the header, there's a GitHub icon with 'v3.2' and '5.8k' stars, and a search bar. Below the header, there are two main examples:

- Add an icon to the map that was generated at runtime.** This example shows a world map with a color-coded heatmap overlay. A specific area in Africa is highlighted with a gradient from green to red. Below the map, the text reads: "Add an icon to the map that was generated at runtime."
- Generate and add a missing icon to the map.** This example shows a world map with several colored squares overlaid on specific countries: a large orange square over South America, a red square over Africa, and a yellow square over Asia. Below the map, the text reads: "Generate and add a missing icon to the map."
- Dynamically generate a missing icon at runtime and add it to the map.** This example shows a world map with a speech bubble containing text: "One longer line". Three lines labeled "Line 1", "Line 2", and "Line 3" are drawn from the speech bubble to different parts of the map. Below the map, the text reads: "Dynamically generate a missing icon at runtime and add it to the map."
- Add a stretchable image to the map.** This example shows a world map with a large, semi-transparent watermark-like image of a map of North America and Europe. The image is stretched to cover a significant portion of the map. Below the map, the text reads: "Add a stretchable image to the map."

On the left side of the page, there's a sidebar with a vertical list of documentation categories:

- Examples
- Overview
- Display buildings in 3D
- Extrude polygons for 3D indoor mapping
- 3D Terrain
- Add a 3D model with babylon.js
- Adding 3D models with three.js on terrain
- Add a 3D model with three.js
- Add a default marker
- Create deck.gl layer using REST API
- Add an animated icon to the map
- Add a generated icon to the map
- Generate and add a missing icon to the map
- Add a stretchable image to the map
- Add an icon to the map
- Animate a line
- Animate map camera around a point
- Animate a series of images
- Animate a marker
- Animate a point
- Animate a point along a route
- Change the default position for attribution
- Customize camera animations
- Add a canvas source
- Center the map on a clicked symbol
- Change building color based on zoom level
- Change the case of labels
- Check if WebGL is supported

On the right side, there's a "Table of contents" sidebar with a long list of topics:

- Display buildings in 3D
- Extrude polygons for 3D indoor mapping
- 3D Terrain
- Add a 3D model with babylon.js
- Adding 3D models with three.js on terrain
- Add a 3D model with three.js
- Add a default marker
- Create deck.gl layer using REST API
- Add an animated icon to the map
- Add a generated icon to the map
- Generate and add a missing icon to the map
- Add a stretchable image to the map
- Add an icon to the map
- Animate a line
- Animate map camera around a point
- Animate a series of images
- Animate a marker
- Animate a point
- Animate a point along a route
- Change the default position for attribution
- Customize camera animations
- Add a canvas source
- Center the map on a clicked symbol
- Change building color based on zoom level
- Change the case of labels
- Check if WebGL is supported
- Display HTML clusters with custom properties

<https://maplibre.org/maplibre-gl-js/docs/>

# MapLibre GL JS

NEW

Documentation

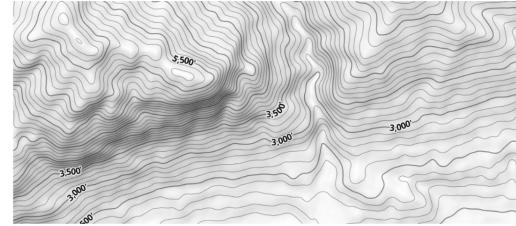
Overview

Introduction Plugins Style Specifications API Examples Guides

GitHub 14.3.2 5.8k 631

Examples  
Overview  
Display buildings in 3D  
Extrude polygons for 3D indoor mapping  
3D Terrain  
Add a 3D model with babylon.js  
Adding 3D models with three.js on terrain  
Add a 3D model with three.js  
Add a default marker  
Create deck.gl layer using REST API  
Add an animated icon to the map  
Add a generated icon to the map  
Generate and add a missing icon to the map  
Add a stretchable image to the map  
Add an icon to the map  
Animate a line  
Animate map camera around a point  
Animate a series of images  
Animate a marker  
Animate a point  
Animate a point along a route  
Change the default position for attribution  
Customize camera animations  
Add a canvas source

## Add Contour Lines



Add contour lines to your map from a raster-dem source.

## Add custom icons with Markers



Add custom marker icons to your map.

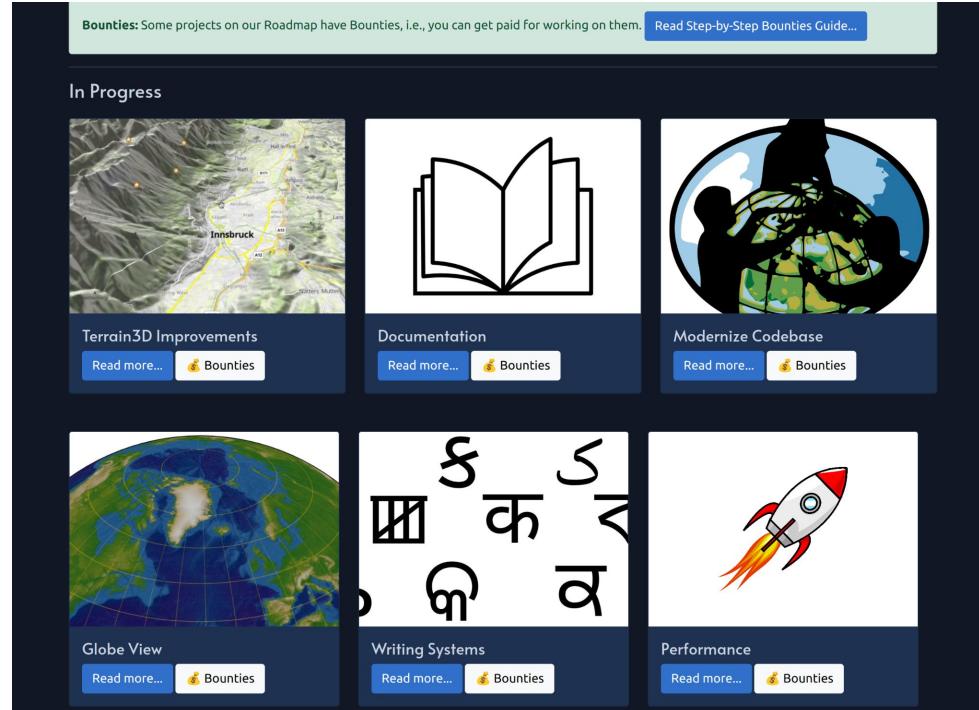
## Table of contents

Change the default position for attribution  
Customize camera animations  
Add a canvas source  
Center the map on a clicked symbol  
Change building color based on zoom level  
Change the case of labels  
Check if WebGL is supported  
Display HTML clusters with custom properties  
Create and style clusters  
Change a layer's color with buttons  
Add Contour Lines  
Cooperative gestures  
Add custom icons with Markers  
Add a custom style layer  
Style lines with a data-driven property  
Disable map rotation  
Disable scroll zoom  
Display and style rich text labels  
Create a draggable Marker  
Create a draggable point  
Use a fallback image  
Add a pattern to a polygon  
Filter symbols by text input  
Filter symbols by toggling a list

# MapLibre GL JS

Bounties 💰 💰

<https://maplibre.org/roadmap/>



<https://maplibre.org/roadmap/step-by-step-bounties-guide/>

# MapLibre GL JS

Sponsors  

<https://maplibre.org/sponsors/>

Current Members of the Sponsorship Program

Gold Tier



Silver Tier





Donations by Individuals

If you personally want to support MapLibre with a donation you have two options:

- Either you can make a donation on GitHub: <https://github.com/sponsors/maplibre/>
- Or you can make a donation on OpenCollective: <https://opencollective.com/maplibre/>

# MapLibre GL JS

Última hora

<https://maplibre.org/jobs/>

- Coordinator
- Graphics Engineer - MapLibre GL JS

## MapLibre Tile Format - MLT

Nuevo formato vector tiles  
(experimental)

Sufijo .mlt

6x reducción de tamaño vs MVT

Expresión `distance`, devuelve la distancia más corta en metros entre la entidad evaluada y la geometría de entrada.



```
"some-property": [ "distance", { "type": "Point", "coordinates": [ 0, 0 ] } ]
```

# MapTiler SDK JS

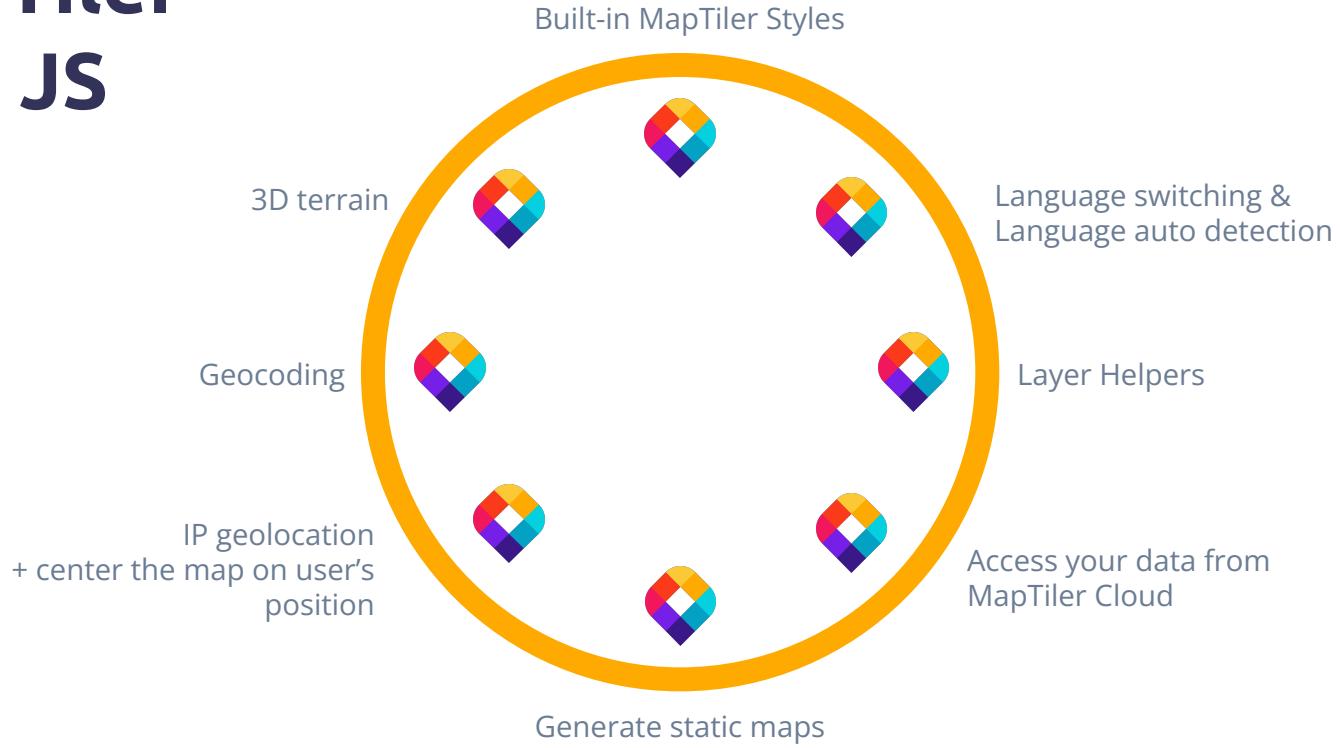


- Una extensión de **MapLibre GL JS**
- JavaScript / TypeScript [open source library](#) (BSD-3)
- Acceso a la API del MapTiler Cloud (geocoding, geolocation, transformación de coordenadas)
- Terreno 3D, Weather, AR Control, Profile Control, Layer Helpers, etc.
- No vendor lock-in (tilesets de cualquier proveedor)

# MapTiler SDK JS

- Una extensión de **MapLibre GL JS**
- JavaScript / TypeScript [open source library](#) (BSD-3)
- Acceso a la API del MapTiler Cloud (geocoding, geolocation, transformación de coordenadas)
- Terreno 3D, Weather, AR Control, Profile Control, Layer Helpers, etc.
- No vendor lock-in (tilesets de cualquier proveedor)

# MapTiler SDK JS



# MapTiler SDK JS

En tu proyecto JS/TS con NPM

Instalación

```
# Install MapTiler SDK
npm install @maptiler/sdk
```

En un proyecto con vanilla JS/HTML desde CDN

```
<!-- in the <head> of HTML document -->
<script src="https://cdn.maptiler.com/maptiler-sdk-js/latest/maptiler-sdk.umd.min.js"></script>
<link href="https://cdn.maptiler.com/maptiler-sdk-js/latest/maptiler-sdk.css" rel="stylesheet" />
```

# MapTiler SDK JS

Objeto de  
configuración

```
import * as maptilersdk from '@maptiler/sdk';

maptilersdk.config.apiKey = "YOUR_MAPTILER_API_KEY_HERE";
maptilersdk.config.primaryLanguage = maptilersdk.Language.SPANISH;
maptilersdk.config.caching = false; //Default true
maptilersdk.config.session = false; //Default true
```

# MapTiler SDK JS

Opciones del  
mapa



Nuevas  
opciones  
!

```
import { Map, MapStyle, Language } from "@maptiler/sdk";

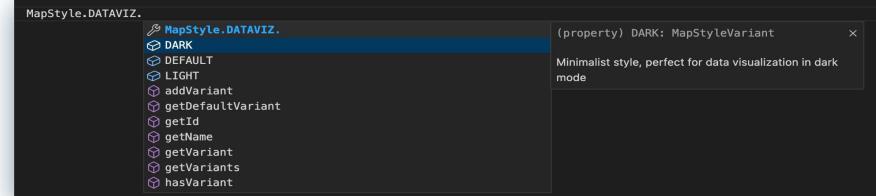
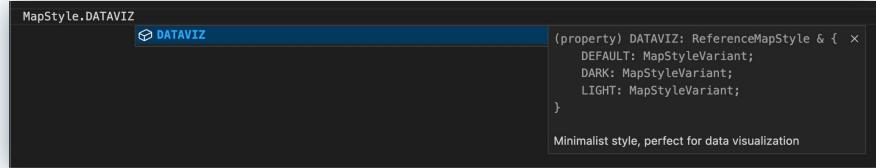
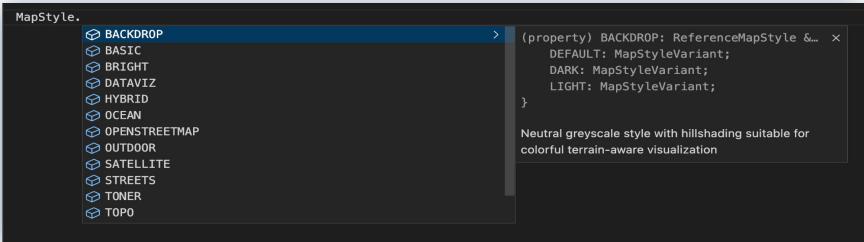
const map = new Map({
  container: "map-container",
  style: MapStyle.WINTER,           // 30+ mapTiler styles, plus your customs ones
  terrain: true,                  // Activate 3D terrain (default exaggeration: 1)
  terrainExaggeration: 1.5,        // Bigger terrain!
  language: Language.JAPANESE,     // 50+ languages, with fallback to local names
  geolocate: true,                // Init the map on the end user location (IP)
  scaleControl: true,             // Add a scale
  terrainControl: true,           // Add a button to toggle the terrain
  geolocateControl: true,         // Add a button to geolocate with GPS
  fullscreenControl: true,          // Add a button to display the map fullscreen
});
```

# MapTiler SDK JS

## Estilos y variantes

¡Nuestro equipo de cartógrafos crea y mantiene muchos estilos de mapas, adecuados para todo tipo de usos!

Cada uno está muy bien documentado y es accesible con el autocompletado VSCode IntelliSense.



# MapTiler SDK JS

Estilos y variantes



# MapTiler SDK JS

Define y actualiza el  
estilo de tu mapa

```
// Set style from the constructor
const map = new Map({
  container,
  style: MapStyle.OUTDOOR,
});

// Update it later, will autocomplete nicely
map.setStyle(MapStyle.BACKDROP);

// You can also use the style ID
map.setStyle("streets-v2");

// Especially useful for your custom styles
map.setStyle("c912fffc8-2360-487a-973b-59d037fb15b8");

// Still possible to do it with a full URL
// (API key necessary only if different than the one you set in the config)
map.setStyle("https://api.maptiler.com/maps/winter-v2/style.json?key=API_KEY");

// You may want to dynamically create a style and pass the JS object
map.setStyle({id: "custom_style", sources: { ... }, layers: [ ... ], ...});
```

# MapTiler SDK JS

## Idiomas

El idioma *fallback* es siempre el idioma local, aka. `{name}`

```
// Set language from the constructor
const map = new Map({
  container,
  language: Language.ENGLISH, // Force the map to use English labels
});

// Update the language later on
map.setLanguage(Language.ALBANIAN);

// You can also use the 2-letter iso code
map.setLanguage("sq");

// Special non-language flags exist:
Language.AUTO      // [DEFAULT] uses the language defined in the web browser

Language.LOCAL     // Uses the language local to each country

Language.LATIN      // Uses a default with latin characters

Language.NON_LATIN // Uses a default with non-latin characters

Language.STYLE_LOCK // to strictly use the language defined in the style.
                     // Prevents any further language update
```

# MapTiler SDK JS

## Idiomas

Incluye soporte para idiomas "right-to-left" (Hebrew, Arabic, etc) sin necesidad de ningún plugin.



# MapTiler SDK JS

Terreno 3D



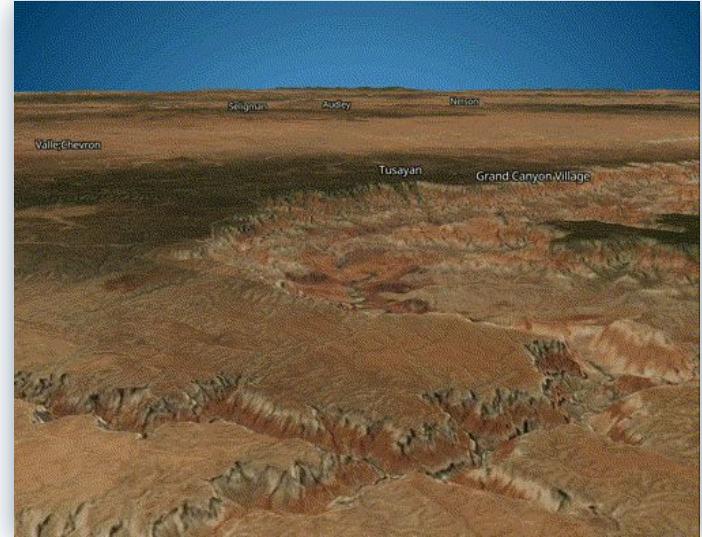
```
const map = new maptilesdk.Map({
  container: 'map', // container's id or
  style: maptilesdk.MapStyle.STREETS,
  center: [8.5419, 47.3799], // starting
  zoom: 10.68, // starting zoom
  terrain: true,
  terrainControl: true,
});
```

MapLibre

```
map.on('style.load', () => {
  map.addSource('terrainSource', {
    'type': 'raster-dem',
    'url': `https://api.maptiler.com/tiles/terrain-rgb-v2/tiles.json?key=${key}`,
    'tileSize': 512,
    'maxzoom': 14
  });
  // add the DEM source as a terrain layer with exaggerated height
  map.setTerrain({ 'source': 'terrainSource', 'exaggeration': 1.5 });
  // add the TerrainControl
  map.addControl(
    new maplibregl.TerrainControl({
      source: 'terrainSource',
      exaggeration: 1
    })
  );
});
```

# MapTiler SDK JS

Terreno 3D



# MapTiler SDK JS

## Otras APIs



<https://docs.maptiler.com/client-js/>

```
● ● ●

import { config, geocoding, geolocation, staticMaps, data, coordinates } from "@maptiler/sdk";

// Set your MapTiler Cloud API key
config.apiKey = "YOUR_API_KEY";

// Look for a city, a country, an address, etc.
const results = await geocoding.forward("Prizren");

// Reverse geocoding + many options. Result as GeoJSON
// Get a place info from [lng, lat]
const resultRGC = await geocoding.reverse([6.249638, 46.402056]);

// Get your own data stored on MapTiler Cloud (GeoJSON)
const resultData = await data.get("2df5ecc4-3ae1-4d1e-99a2-182256486387");

// IP geolocation
// Get place info of an end user: country, city, timezone, language spoken in area + more
const resultGL = await geolocation.info()

// Transform coordinates from one EPSG to another
// EPSG.io vibes, but in JS/TS!
// Here converting [lng, lat] from EPSG:4326 (WGS84, the default) to Lambert-93
const resultTransf = await coordinates.transform([1, 45], {targetCrs: 9793});

// Static map, generates a link. Many options, you can add paths and markers
const imageLinkCentered = staticMaps.centered([7, 46.8], 5, options);
```



# MapTiler SDK JS

Ejemplos de mapas estáticos



# MapTiler SDK JS

Layer Helpers Puntos



```
import { helpers } from "@maptiler/sdk";

helpers.addPoint(map, {
  data: "public-schools.geojson",
  pointColor: "#F1175D",
});
```

📍 MapLibre



```
map.addSource('point', {
  'type': 'geojson',
  'data': data
});

map.addLayer({
  'id': 'point',
  'source': 'point',
  'type': 'circle',
  'paint': {
    'circle-radius': 10,
    'circle-color': '#007cbf'
  }
});
```

# MapTiler SDK JS

## Layer Helpers Puntos



```
helpers.addPoint(map, {
  data: "public-schools.geojson",
  property: "students",
  pointColor: ColorRampCollection.PORTLAND.scale(200, 2000).resample("ease-out-sqrt"),
  pointOpacity: 0.8,
  minPointRadius: 6,
  maxPointRadius: 30,
  showLabel: true,
  zoomCompensation: false,
})
```



# MapTiler SDK JS

## Layer Helpers Cluster



```
await maptilersdk.helpers.addPoint(map, {  
  data: 'schools.geojson',  
  cluster: true,  
});
```

<https://docs.maptiler.com/sdk-js/api/helpers/>

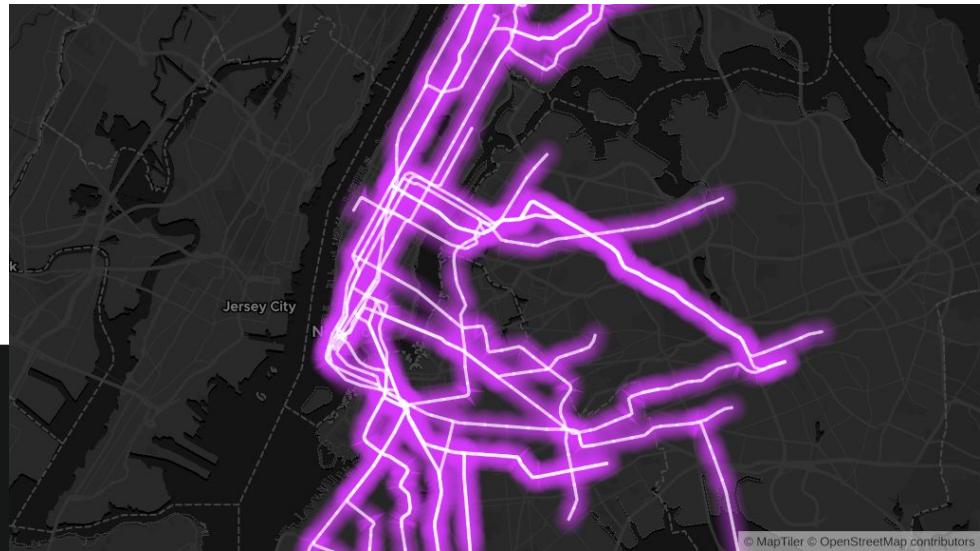


```
map.addSource('earthquakes', {  
  type: 'geojson',  
  data: 'earthquakes.geojson',  
  cluster: true,  
  clusterMaxZoom: 14,  
  clusterRadius: 50  
});  
  
map.addLayer({  
  id: 'clusters',  
  type: 'circle',  
  source: 'earthquakes',  
  filter: ['has', 'point_count'],  
  paint: {  
    'circle-color': [  
      'step',  
      ['get', 'point_count'],  
      '#51bbd6',  
      100,  
      '#f1f075',  
      750,  
      '#f28cb1'  
    ],  
    'circle-radius': [  
      'step',  
      ['get', 'point_count'],  
      20,  
      100,  
      30,  
      750,  
      40  
    ]  
  }  
});  
  
map.addLayer({  
  id: 'cluster-count',  
  type: 'symbol',  
  source: 'earthquakes',  
  filter: ['has', 'point_count'],  
  layout: {  
    'text-field': '{point_count_abbreviated}',  
    'text-font': ['DIN Offc Pro Medium'],  
    'text-size': 12  
  }  
});  
  
map.addLayer({  
  id: 'unclustered-point',  
  type: 'circle',  
  source: 'earthquakes',  
  filter: ['!', ['has', 'point_count']],  
  paint: {  
    'circle-color': '#11b4da',  
    'circle-radius': 4,  
    'circle-stroke-width': 1,  
    'circle-stroke-color': '#fff'  
  }  
});
```

# MapTiler SDK JS

## Layer Helpers Lines

```
● ● ●  
  
await maptilersdk.helpers.addPolyline(map, {  
  data: 'YOUR_MAPTILER_DATASET_ID_HERE',  
  // --> Main line  
  lineColor: "#fff",          // White  
  lineOpacity: 0.7,           // Partially transparent to see the halo color through  
  
  // --> Halo line  
  outline: true,              // To create the electric halo, the outline is  
  outlineColor: "#d83bff",    // purple  
  outlineOpacity: 0.5,         // half transparent  
  outlineWidth: 15,           // quite large compared to the main line  
  outlineBlur: 15,             // and blurry  
});
```

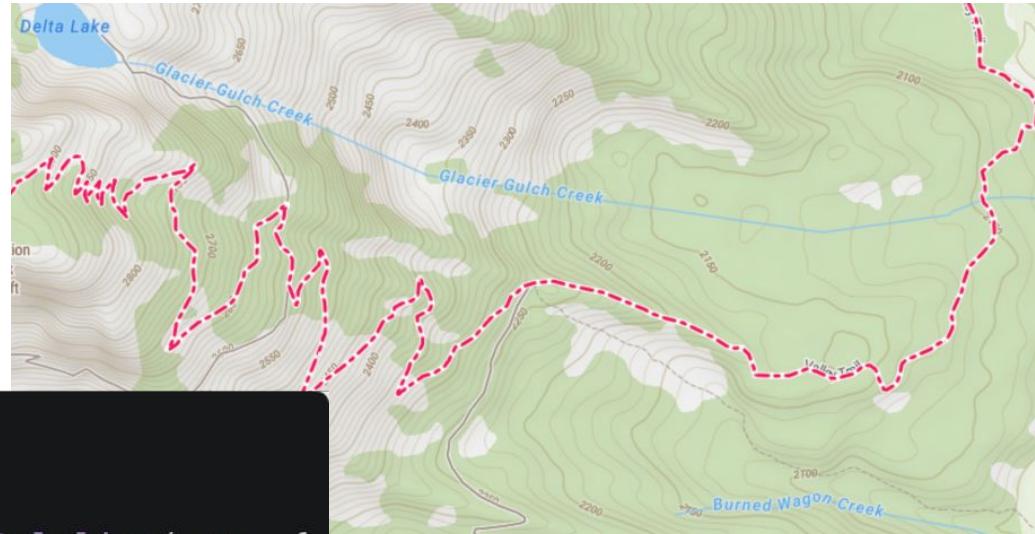


# MapTiler SDK JS

## Layer Helpers Lines

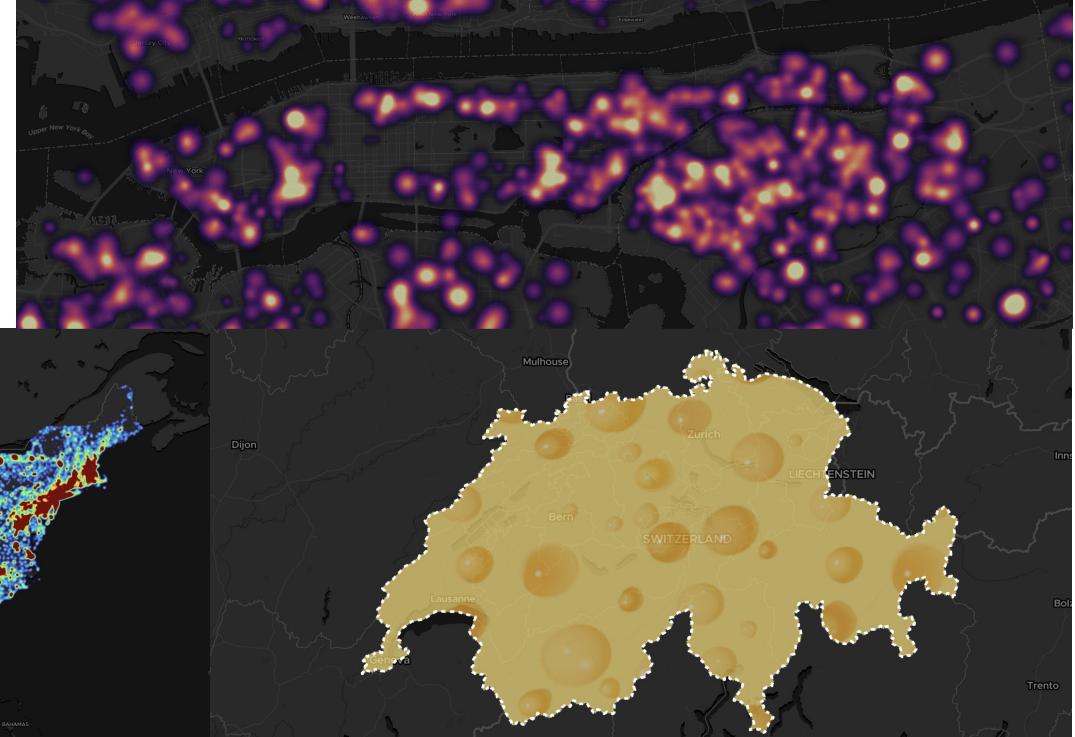


```
await maptilersdk.helpers.addPolyline(map, {  
  data: 'YOUR_MAPTILER_DATASET_ID_HERE',  
  outline: true,  
  lineDashArray: "___ _ ", //or [3,2,1,2]  
  lineColor: "#F1175D"  
});
```



# MapTiler SDK JS

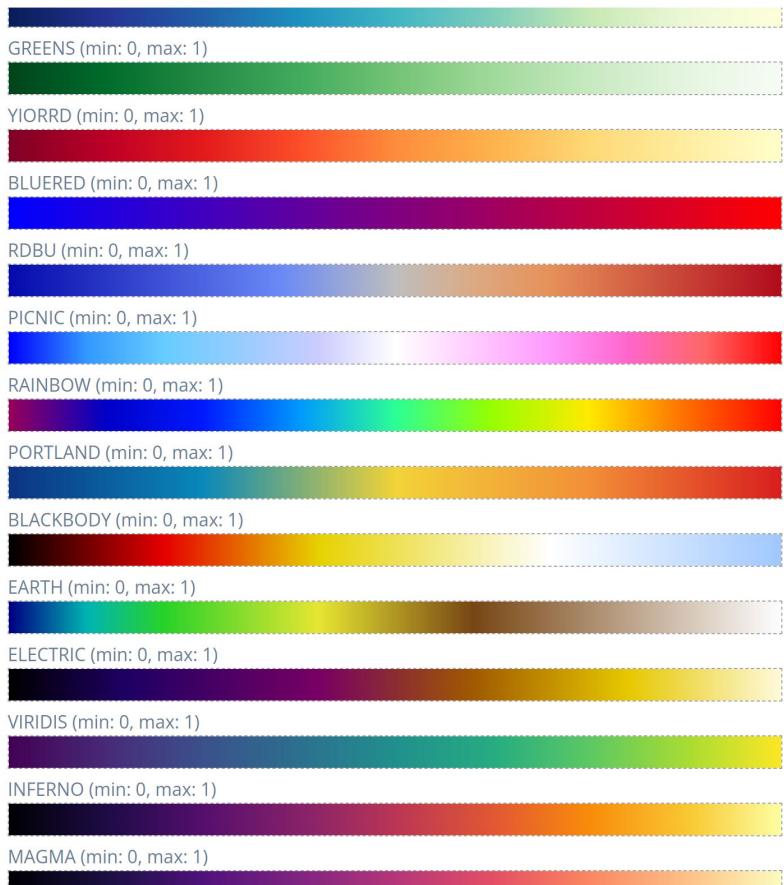
Layer Helpers: Polígonos,  
Heatmaps



# MapTiler SDK JS

## Color ramps

```
import { ColorRamp, ColorRampCollection} from "@maptilesr/sdk";  
  
//use a built-in color ramp  
ColorRampCollection.WARM  
  
const myCustomRamp = new ColorRamp({  
  stops: [  
    { value: 0, color: [68, 1, 84] },  
    { value: 13, color: [71, 44, 122] },  
    { value: 25, color: [59, 81, 139] },  
    { value: 38, color: [44, 113, 142] },  
    { value: 5, color: [33, 144, 141] },  
    { value: 63, color: [39, 173, 129] },  
    { value: 75, color: [92, 200, 99] },  
    { value: 88, color: [170, 220, 50] },  
    { value: 100, color: [253, 231, 37] }  
  ]  
});
```



# MapTiler SDK JS

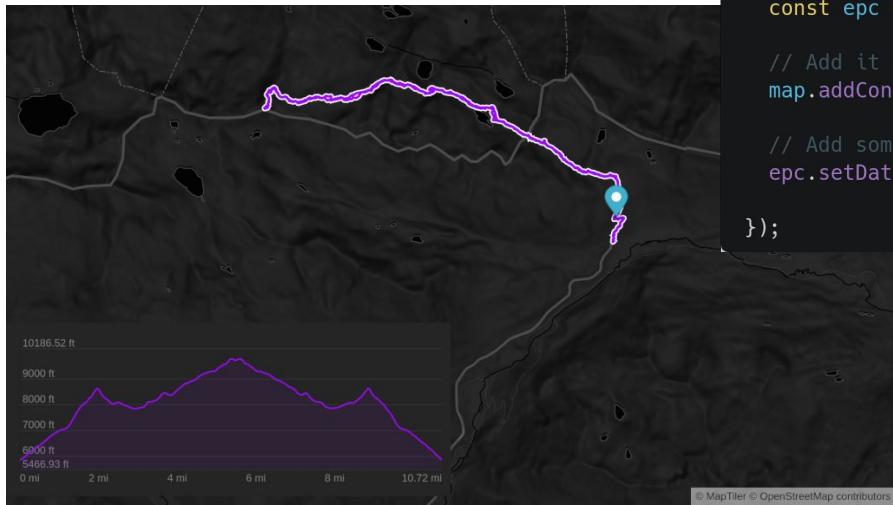
## Módulos externos

Estamos construyendo un ecosistema  
alrededor de MapTiler Cloud y MapTiler SDK



# MapTiler SDK JS

## Profile control



```
● ● ●  
import { ElevationProfileControl } from "@maptiler/elevation-profile-control";  
  
map.on('load', () => {  
  
    // Create an instance (with no options)  
    const epc = new ElevationProfileControl();  
  
    // Add it to your map  
    map.addControl(epc);  
  
    // Add some data (from a URL or a MapTiler Data UUID)  
    epc.setData('YOUR_MAPTILER_DATASET_ID_HERE');  
  
});
```



# MapTiler SDK JS

Geocoding control



```
import { GeocodingControl } from "@maptiler/geocoding-control/maptilersdk";
const gc = new GeocodingControl({});
map.addControl(gc);
```



<https://docs.maptiler.com/sdk-js/modules/geocoding/>



# MapTiler SDK JS

AR control



```
import {MaptilerARControl} from "@maptiler/ar-control";

map.on('load', function () {
    // Creating the control
    const arControl = new MaptilerARControl();

    // Adding the AR control on the top-left corner of the map
    map.addControl(arControl, "top-left");
});
```

# MapTiler SDK JS

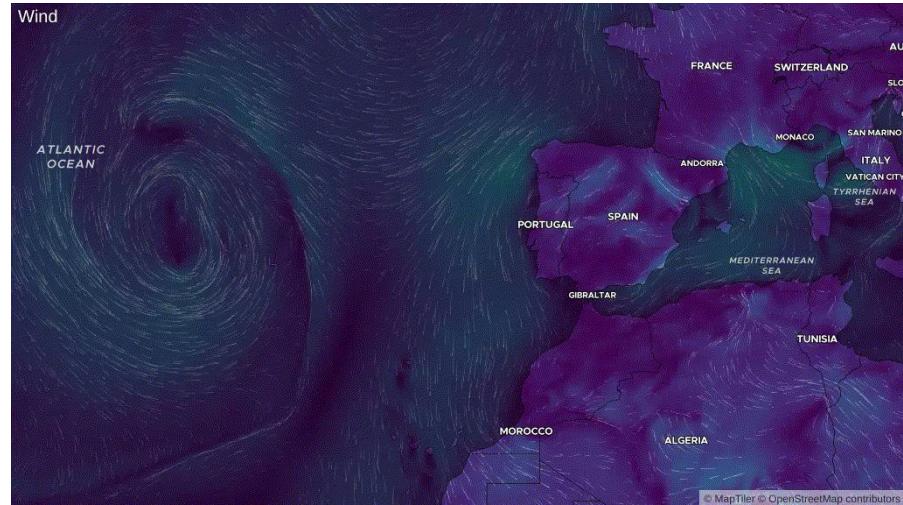
Weather JS 

## Module Reference

- Precipitation layer
- Pressure layer
- Radar layer
- Temperature layer
- Wind layer
- Layers methods
- Layers events
- Color ramp
- Pressure isolines (plus)
- Wind arrow (plus)

## Low-level Reference

- Intensity layer
- ColoringFragment
- Particle layer
- Arrow layer (plus)
- Isolines layer (plus)
- Types and interfaces



```
import { WindLayer } from "@maptiler/weather";
const windLayer = new WindLayer();
```

# MapTiler

## SDK JS + *Leaflet*

### Leaflet JS con Vector Tiles

- Mapas base de vector tiles multilingües para Leaflet utilizando el SDK de MapTiler.
- Capas de vector tiles (data overlay) en tus aplicaciones sin ningún tipo de limitación (mostrar millones de geometrías, mapas de coropletas, etc).
- Cambiar fácilmente el idioma del mapa sin tener que crear un nuevo mapa base.
- Utiliza cualquiera de los mapas base creados por MapTiler o utiliza un mapa base personalizado.

```
// Import the Leaflet MapTiler Plugin
import "@maptiler/leaflet-maptilersdk";

const map = L.map('map', {
  center: L.latLng(49.2125578, 16.62662018),
  zoom: 14,
});

// Create a MapTiler Layer inside Leaflet
const mtLayer = new L.MaptilerLayer({
  // Get your free API key at https://cloud.maptiler.com
  apiKey: "YOUR_MAPTILER_API_KEY_HERE",
}).addTo(map);
```

# MapTiler SDK JS

## Documentación

**Examples**  
Explore our code samples collection.

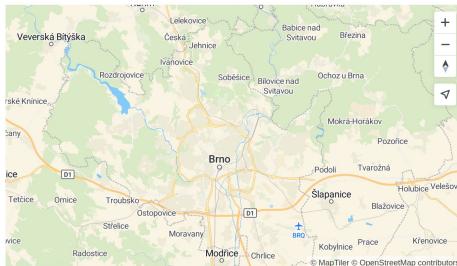
[EXPLORE THE EXAMPLES](#)

**SDK JS Reference**  
Read the SDK API Reference.

[READ DOCUMENTATION](#)

### Get started integrating maps

Simply use the code below the map.



### Video tutorials



[More video  
tutorials](#)

### JS Frameworks

[Angular](#)  
[React](#)  
[Svelte](#)  
[Vue.js](#)  
[Vite](#)



### Add custom icon (SVG) to a point layer

[Example](#)

This example shows how to make a map with pins to display a point layer from a MapTiler Cloud Tileset using a custom SVG icon.



### Display GPX track elevation profile

[Example](#)

The elevation profile control for MapTiler SDK is a super easy way to show the elevation profile of any GPX track, with elevation data fueled by MapTiler Cloud.



### Elevation at point

[Example](#)

How to get the elevation in meters from any location.



### Map view URL hash

[Example](#)

Share and retrieve a particular map view by simply copying and pasting the URL. Maintain the map view in the URL hash.



Gracias!  
Preguntas?



X @bolosig

in bolosig

maptiler



<https://github.com/maptiler>