پارسی‌مپ

www.parsimap.com

عنوان مستند:

مستند بکارگیری نقشه در وب

(نسخه 3.1.0)

شرکت گسترش فاوای دانش شهر

1398/04/15

فهرست

نصب و راه‌اندازی

تنظیمات اولیه نقشه

ایجاد و به‌کارگیری نقشه پارسی‌مپ نیاز به تخصیص API Key معتبر دارد. این مقدار مجوزی برای استفاده از نقشه و پیاده‌سازی اجزا مختلف ازجمله تصاویر نقشه (Tile) دارد. برای افزودن اسکرپیت اصلی نقشه می‌بایست تگ script که حاوی آدرس فایل parsimap.js است درون بخشی از document صفحه قرار گیرد. درصورتی‌که جایگاه قرارگیری اسکریپت درون تگ head باشد برای استفاده از کلاس‌های نقشه دیگر نیازی به قرارگیری کدهای نقشه درون رخداد ready صفحه نخواهد بود و مستقیماً می‌توان از آن‌ها بهره برد.

```
<head>
  <script
    src="https://www.parsimap.com/js/v3.1.0/parsimap.js?key=API_KEY"></script>
</head>
```

در کد فوق جایگاه قرارگیری script درون تگ head است و در آن پارامتر QueryString مربوط به key حاوی عبارت API_KEY است که این عبارت باید با مقداری که بیان‌گر API Key نقشه است جایگزین شود.

```
<head>
  <script src="https://www.parsimap.com/js/v3.1.0/parsimap.js?key=public"></script>
</head>
```

استفاده از مقدار public در پارامتر key مربوط به QueryString اسکریپت پارسی‌مپ بدین معناست که عمل login یا احراز هویت انجام‌نشده و تنها می‌توان از حداقل سرویس‌های اختصاص‌یافته ازجمله tile نقشه برای نمایش نقشه و اجزاء نقشه استفاده نمود. سایر موارد مربوط به نقشه مانند extenstionsها پس از این‌عمل قابل‌استفاده نخواهند بود.

جایگاه نمایش نقشه در document

نقشه پارسی‌مپ برای قرار گرفتن در document صفحه و نمایش آن لازم است که درون تگ body قرار یافته و تگ مورداستفاده برای آن می‌بایست div باشد. این تگ در حقیقت مکانی است که نقشه در آن بارگذاری شده و به نمایش در می‌آید.

مقداردهی اولیه

مقداردهی اولیه شامل تعریف شیء از کلاس Map و اختصاص مقادیر اولیه به آن است.

تعریف شیء از کلاس Map

Map یک کلاس درون namespace پارسی‌مپ، ارائه‌دهنده یک نقشه بر بستر وب است. شیءهای ایجادشده از این کلاس هرکدام دارای Instance متفاوت از هم بوده و این امکان باعث می‌شود تا نقشه‌های متعددی را بروی یک صفحه ایجاد نمود.

```
var map = new parsimap.Map(MAP_CONTAINER, { MAP_OPTIONS });
```

شکل کلی استفاده از این کلاس به‌صورت بالا است به‌طوری‌که یک شیء map از کلاس Map در فضای نامی parsimap با مقادیر اولیه MAP_OPTIONS و MAP_CONTAINER ایجاد می‌گردد.

ایجاد نقشه

هر نقشه برای ایجاد شدن حداقل نیازمند تعیین سه گزینه دارد، این گزینه‌ها شامل مواردی ازجمله container (جایگاه قرارگیری المنت نقشه)، center (مختصات اولیه مرکز نقشه) و zoom (سطح بزرگ‌نمائی اولیه نقشه) است.

```
var container = document.getElementById("map");
var mapOptions = {
  zoom: 18,
  center: [51.41, 35.7575]
};
var map = new parsimap.Map(container, mapOptions);
```

در کد فوق متغیر container حاوی تگ div با شناسه map است که از این مقدار به‌عنوان جایگاه قرارگیری المنت نقشه استفاده کرد، این تگ باید با توجه به توضیحات مربوط به بخش جایگاه نمایش نقشه در document قبلاً به document افزوده‌شده باشد. متغیر mapOptions شامل دو گزینه zoom و center است که به ترتیب مقادیر اولیه را برای سطح بزرگنمایی و مختصات اولیه مرکز نقشه تعیین می‌کنند. متغیر container به‌عنوان اولین آرگومان ورودی تابع سازنده یا constructor کلاس Map در نظر گرفت شده و سایر گزینه‌ها به‌عنوان گزینه‌های نقشه درون دومین آرگومان تابع سازنده تخصیص می‌یابند.

تعیین سبک نقشه

سایر گزینه‌های نقشه می‌تواند شامل **style** است که از آن برای تعیین استایل اولیه نقشه استفاده می‌گردد. مقدار اولیه آن برابر **parsimap://map** است که بیانگر تصاویر **raster** نقشه است، همچنین می‌توان از تصاویر **raster** ماهواره‌ای هم در نقشه استفاده که برای این کار می‌بایست عبارت **parsimap://sat** درون گزینه **style** قرار یابد، همچنین برای استفاده از تصاویر **vector** نقشه باید عبارت **parsimap://street** قرار یابد.

```
var container = document.getElementById("map");
var mapOptions = {
  center: [51.41, 35.7575],
  zoom: 18,
  style: "parsimap://street"
};
var map = new parsimap.Map(container, mapOptions)
```

در کد فوق **style** که یکی از گزینه‌های نقشه یا دومین ارگومان ورودی تابع سازنده Map برابر عبارت **parsimap://street** قرارگرفته که منظور از آن تعیین سبک اولیه نقشه با استفاده از تصاویر **vecotr** نقشه است.

تعیین ارتفاع المنت نقشه

برای نمایش یافتن نقشه نیاز به تعیین خصوصیت **height** استایل تگ **div** که جایگاه قرارگیری نقشه است، خواهد بود.

```
<style>
  body,
  html {
    height: 100%;
  }
  #map {
    height: 100%;
  }
</style>
<body>
  <div id="map"></div>
</body>
```

در کد فوق نقشه به ازای 100% نمای جاری ظاهر خواهد شد به‌طوری‌که مقدار height را از المنت والد خود به‌صورت اندازه کامل می‌گیرد. عدم تعیین این خصوصیت منجر به این می‌شود که نقشه در صفحه ظاهر نشده و قابل‌استفاده نباشد. در این نمونه والد المنت نقشه تگ body است ولی درصورتی‌که والد المنت نقشه توالی از تگ ها باشد باید به ازای هر والد تا body ارتفاع 100% یا در مقداری خاص تعیین‌شده باشد چراکه بر اساس وراثت ارتفاع، هر تگ از والد خود مقدار height را به ارث می‌برد ازاین‌رو اگر ارتفاع تگ والد برابر 40 پیکسل باشد و ارتفاع نقشه برابر 50% باشد بر اساس ارث‌بری طبیعتاً باید ارتفاع نقشه 20 پیکسل باشد.

نوع داده‌ها

LngLat

- Array<number, number>

نوع مختصات جغرافیایی از یک آرایه با دو عضو تشکیل‌شده است، عضو اول آن مربوط به **longitude** یا عرض جغرافیائی و عضو دوم آن مربوط به **latitude** طول جغرافیائی است. مقادیر داخل هر عضو می‌توان از نوع عدد صحیح و اعشار باشد.

```
var lngLat = [51.41, 35.7575];
```

در نمونه کد عرض جغرافیائی برابر مقدار **51.41** و طول برابر **35.7575** است.

LngLatBounds

- Array<LngLat, LngLat>

ImageLike

- string name
- string url

ImageCollection

- Array<ImageLike>

نوع مجموعه تصاویر از آرایه‌ای از اسامی و آدرس‌های تصویر تشکیل‌شده است به‌طوری‌که هر عضو از آرایه شامل **name** به معنی نام تصاویر جهت استفاده در نقشه و **url** آدرس فیزیکی تصویر است که باید برای نمایش آن در نقشه تعیین‌شده باشد.

```
var imageCollection = [
  { name: "drone", url: "HOST_URL/images/drone.png" },
  { name: "solar", url: "HOST_URL/images/solar.png" }
];
```

در نمونه کد دو عضو با محتوای مدنظر از نوع ImageCollection قرار دارند به‌طوری‌که در فیلد **url** هر عضو یک آدرس واقعی یا فیزیکی از تصویر قرار یافته که بر اساس آن تصویر به نقشه افزوده خواهد شد، همچنین **HOST_URL** آدرس نمادین هاست مبدأ است.

Map

Constructor

`Map(container, options)`

پارامترها:

- `Element container`
- `MapOptions options`

یک نقشه درون المنت **container** که معمولاً یک تگ **div** است، ایجادشده و در **document** نشان داده می‌شود.

MapOptions

- `LngLat center` – مختصات جغرافیایی اولیه مرکز نقشه.
- `number zoom` – سطح بزرگ‌نمائی اولیه نقشه.
- `string style` (غیرضروری) – سبک اولیه نقشه. پیش‌فرض: **parsimap://map**.

متدهای نقشه

متدها با نقشه تعامل داشته و عملی خاص را بروی آن انجام می‌دهند، در نمونه‌های زیر فرض براین است که map یک instance از کلاس Map است، که در بخش تنظیمات اولیه نقشه به‌طور کامل شرح داده شد.

map.on(eventName, handler)

پارامترها:

- string eventName
- Function layerHandler

مقدار بازگشتی: number

توسط متد on می‌توان رخداد خاصی را به نقشه افزود. پارامتر اولیه آن eventName شامل نام رخداد مدنظر است که در بخش رخدادهای نقشه فهرست آن‌ها شرح داده‌شده است. پارامتر دوم handler می‌تواند حاوی function باشد که هنگام صدا زده شدن event تابع مذکور نیز با ارگومان‌های مرتبط فراخوانی می‌گردد.

map.on(eventName, layerId, layerHandler)

پارامترها:

- string eventName
- string layerId
- Function layerHandler

مقدار بازگشتی: number

این متد درصورتی‌که با سه پارامتر مورداستفاده قرار گیرد ساختار پارامترها بدین شکل تغییر می‌کند. پارامتر eventName نام رخداد مربوط به layer مدنظر است که در بخش رخدادهای لایه مورد بررسی قرار می‌گیرد. دومین پارامتر آن layerId است که بیانگر شناسه مربوط به لایه‌ای خواهد بود که نیاز است رخدادی به آن فزوده شود. layerHandler می‌تواند حاوی function باشد که هنگام صدا زده شدن event تابع نیز با ارگومان‌های مرتبط فراخوانی می‌گردد.

map.off(eventName, oldHandler)

پارامترها:

- `string eventName`
- `Function oldHandler`

مقدار بازگشتی: فاقد مقدار

با توجه به `handler` که قبلاً توسط متد `on` به یکی از رخدادهای نقشه اختصاص‌یافته، `function` حذف می‌شود. اولین پارامتر این متد شامل `eventName` است که برگیرنده نام رخداد است و `oldHandler` شامل `handler` است که هنگام افزودن رخداد به نقشه توسط متد `map.on` به یکی از رخدادهای نقشه اختصاص یافته است و با تخصیص مجدد تابع قبلی، رخداد مدنظر از نقشه حذف می‌گردد.

map.off(eventName, layerId, oldLayerHandler)

پارامترها:

- `string eventName`
- `string layerId`
- `Functin oldLayerHandler`

مقدار بازگشتی: فاقد مقدار

درصورتی‌که با سه پارامتر مورداستفاده قرار گیرد ساختار پارامترها بدین شکل تغییر می‌کند. پارامتر `eventName` نام رخداد مربوط به `layer` مدنظر است که در بخش رخدادهای لایه مورد بررسی قرار می‌گیرد. دومین پارامتر آن `layerId` است که بیانگر شناسه مربوط به لایه‌ای خواهد بود که نیاز است که `function` مربوط به آن حذف گردد، `oldLayerHandler` شامل `handler` است که هنگام افزودن رخداد به لایه توسط متد `map.on` به یکی از رخدادهای لایه اختصاص یافته است و با تخصیص مجدد تابع قبلی، رخداد مدنظر از `layer` حذف می‌گردد.

map.loadImageCollection(imageCollection, handler?)

پارامترها:

- `ImageCollection[] imageCollection`
- `Function handler` (غیرضروری)

مقدار بازگشتی: فاقد مقدار

یک یا مجموعه‌ای از تصاویر را بارگذاری کرده و در نقشه بکار گرفته می‌شود. پارامتر **imageCollection** دربرگیرنده یک یا مجموعه‌ای از تصاویر با نام تصویر است که باید از نوع **ImageCollection** تعریف شده باشد. **Handler** دربرگیرنده تابعی است که پس از بارگذاری و به‌کارگیری تصاویر در نقشه صدا زده می‌شود در بخش امادەسازی تصاویر (پیش‌نیاز لایه **symbol**) به‌طور کامل شرح داده خواهد شد.

map.addSource(id, data, options?)

پارامترها:

- string id
- String|Array|Object data
- Object options (غیرضروری)
  - string type
  - Object cluster
    - number maxZoom
    - number radius

مقدار بازگشتی: فاقد مقدار

یک منبع داده‌های جغرافیائی به نقشه جهت استفاده، افزوده می‌شود. پارامتر **id** دربرگیرنده شناسه منبع است. **data** شامل اطلاعات جغرافیایی است که با توجه به نوع که در **options** می‌بایست اختصاص گردد، انتخاب می‌گردد. برای نمونه می‌توان از نوع‌های **GPF**، **TSV**و **GeoJSON** استفاده نمود. شرح کامل افزودن و استفاده از منابع داده‌های جغرافیایی در بخش افزودن منبع قرار دارد. پارامتر **options** غیرضروری بوده و مقدار پیش‌فرض آن برابر **{ type: 'geojson' }** است. همچنین در گزینه‌های منبع می‌تواند تعیین نمود که امکان بهره‌مند شدن منبع از قابلیت خوشه‌بندی فراهم شود که این کار توسط تعیین گزینه **cluster** امکان‌پذیر خواهد بود.

map.getSource(id, options?)

پارامترها:

- string id
- options (غیرضروری)
  - string type

مقدار بازگشتی:

- object|string data
- Function setData

مقدار بازگشتی حامل **data** و تابع **setData** است که منظور از **data** همان منبع داده‌های جغرافیایی است که قبلاً توسط متد **addSource** به نقشه افزوده‌شده است و توسط **setData** می‌توان داده‌هایی را در منبع تعیین نمود. پارامتر **id** دربرگیرنده شناسه بوده و **options** برای انتخاب **type** منبع استفاده می‌گردد و یک پارامتر غیرضروری است که مقدار پیش‌فرض آن برابر **{ type: 'geojson' }** است. برای نمونه می‌توان از نوع‌های **GPF**، **TSV**و **GeoJSON** استفاده نمود. شرح کامل افزودن و **استفاده** از منابع داده‌های جغرافیایی در بخش افزودن منبع قرار دارد.

map.removeSource(id)

پارامترها:

- string id

مقدار بازگشتی: فاقد مقدار

جهت حذف یک منبع داده‌های جغرافیائی از نقشه از این متد استفاده می‌شود. پارامتر **id** دربرگیرنده شناسه **source** است.

map.addSymbolLayer(id, sourceId, options?)

پارامترها:

- string id
- string sourceId
- Object options (غیرضروری)

مقدار بازگشتی: فاقد مقدار

لایه‌ای از نوع **symbol** ایجاد می‌گردد. پارامتر **id** حامل شناسه لایه است. **sourceId** دربرگیرنده شناسه مربوط به **source** اصلی که قبلاً توسط **addSource** به نقشه افزوده‌شده باشد. در پارامتر **options** گزینه‌های مربوط به **symbol** تخصیص می‌یابد. توضیحات کامل در مورد نحوه ایجاد این لایه در بخش افزودن لایه **symbol** قرار دارد.

map.addLineLayer(id, sourceId, options?)

پارامترها:

- string id
- string sourceId
- Object options (غیرضروری)

مقدار بازگشتی: فاقد مقدار

لایه‌ای از نوع **line** ایجاد می‌گردد. پارامتر **id** حامل شناسه لایه است. **sourceId** دربرگیرنده شناسه مربوط به **source** اصلی که قبلاً توسط **addSource** به نقشه افزوده‌شده باشد. در پارامتر **options** گزینه‌های مربوط به **line** تخصیص می‌یابد. توضیحات کامل در مورد نحوه ایجاد این لایه در بخش افزودن لایه line قرار دارد.

map.addFillLayer(id, sourceId, options?)

پارامترها:

- string id
- string sourceId
- Object options (غیرضروری)

مقدار بازگشتی: فاقد مقدار

لایه‌ای از نوع **fill** ایجاد می‌گردد. پارامتر **id** حامل شناسه لایه است. **sourceId** دربرگیرنده شناسه مربوط به **source** اصلی که قبلاً توسط **addSource** به نقشه افزوده‌شده باشد. در پارامتر **options** گزینه‌های مربوط به **fill** تخصیص می‌یابد. توضیحات کامل در مورد نحوه ایجاد این لایه در بخش افزودن لایه fill قرار دارد.

map.addCircleLayer(id, sourceId, options?)

پارامترها:

- string id
- string sourceId
- Object options (غیرضروری)

مقدار بازگشتی: فاقد مقدار

لایه‌ای از نوع **circle** ایجاد می‌گردد. پارامتر **id** حامل شناسه لایه است. **sourceId** دربرگیرنده شناسه مربوط به **source** اصلی که قبلاً توسط **addSource** به نقشه افزوده‌شده باشد. در پارامتر **options** گزینه‌های مربوط به **circle** تخصیص می‌یابد. توضیحات کامل در مورد نحوه ایجاد این لایه در بخش افزودن لایه circle قرار دارد.

map.addHeatmapLayer(id, sourceId, options?)

پارامترها:

- string id
- string sourceId
- Object options (غیرضروری)

مقدار بازگشتی: فاقد مقدار

لایه‌ای از نوع **heatmap** ایجاد می‌گردد. پارامتر **id** حامل شناسه لایه است. **sourceId** دربرگیرنده شناسه مربوط به **source** اصلی که قبلاً توسط **addSource** به نقشه افزوده‌شده باشد. در پارامتر **options** گزینه‌های مربوط به **heatmap** تخصیص می‌یابد. توضیحات کامل در مورد نحوه ایجاد این لایه در بخش افزودن لایه heatmap قرار دارد.

map.removeLayer(id)

پارامترها:

- string id

مقدار بازگشتی: فاقد مقدار

لایه‌ای از نقشه حذف می‌گردد. پارامتر **id** شناسه مربوط به لایه‌ای است که باید از نقشه حذف گردد.

map.setLayerLayout(layerId, layout)

پارامترها:

- string layerId
- Object layout

مقدار بازگشتی: فاقد مقدار

خصوصیت‌های `layout` یک `layer` را تعیین می‌کند، به‌طوری‌که مقدار جدید با مقدار قبلی آن جایگزین خواهد شد. پارامتر `layerId` تعیین کنند لایه مدنظری است که باید تغییرات روی آن اعمال شود. `layout` حاوی مقادیری جهت تغییر خصوصیت‌های `layout` در لایه است.

map.setLayerPaint(layerId, paint)

پارامترها:

- `string layerId`
- `Object paint`

مقدار بازگشتی: فاقد مقدار

خصوصیت‌های `paint` یک `layer` را تعیین می‌کند، به‌طوری‌که مقدار جدید با مقدار قبلی آن جایگزین خواهد شد. پارامتر `layerId` تعیین کنند لایه مدنظری است که باید تغییرات روی آن اعمال شود. `paint` حاوی مقادیری جهت تغییر خصوصیت‌های `paint` در لایه است.

map.setLayerFilter(layerId, filter)

پارامترها:

- `string layerId`
- `Array filter`

مقدار بازگشتی: فاقد مقدار

خصوصیت `filter` یک `layer` را تعیین می‌کند، به‌طوری‌که مقدار جدید با مقدار قبلی آن جایگزین خواهد شد. پارامتر `layerId` تعیین کنند لایه مدنظری است که باید تغییرات روی آن اعمال شود. `filter` حاوی مقادیری جهت تغییر خصوصیت `filter` در لایه است.

map.getLayer(id)

پارامترها:

- `string id`

مقدار بازگشتی: `object`

مقدار بازگشتی شامل اطلاعات مربوط لایه است که قبلاً تعیین‌شده است.

map.getDefaultSymbol()

پارامترها: فاقد ورودی

مقدار بازگشتی: Object

مقدار بازگشتی حاوی گزینه‌های مربوط به لایه‌ای از نوع symbol است، این لایه دربرگیرنده مارکری با آیکون قرمز رنگی است که می‌توان از آن به‌عنوان نشان دادن محلی خاص بروی نقشه است.

map.setStyle(style)

پارامترها:

- string style

مقدار بازگشتی: فاقد مقدار

براساس تعیین سبک نقشه می‌توان از تصاویر raster یا vector استفاده نمود. پارامتر style دربرگیرنده نام سبک مدنظر است که با تعیین آن تصاویر مربوطه در نقشه بکار گرفته می‌شود.

map.getContainer()

پارامترها: فاقد ورودی

مقدار بازگشتی: Element

مقدار بازگشتی یک تگ div است که حاوی element های نقشه و به‌عبارتی‌دیگر جایگاه نمایش نقشه در document است.

map.getZoom()

پارامترها: فاقد ورودی

مقدار بازگشتی: number

مقدار بازگشت یافته حاوی سطح بزرگ‌نمائی فعلی نقشه است.

map.setZoom(zoom)

پارامترها:

- number zoom

مقدار بازگشتی: فاقد مقدار

سطح بزرگ‌نمائی نقشه را تعیین می‌کند، پارامتر `zoom` دربرگیرنده عددی برای تنظیم سطح بزرگ‌نمائی نقشه است و می‌تواند به‌صورت اعشار یا عدد صحیح بکار گرفته شود.

map.getCenter()

پارامترها: فاقد ورودی

مقدار بازگشتی: LngLat

مقدار بازگشتی حاوی مختصات جغرافیائی مرکز فعلی نقشه است.

map.setCenter(center)

پارامترها:

- LngLat center

مقدار بازگشتی: فاقد مقدار

مختصات مرکز نقشه را تعیین می‌کند، پارامتر `center` دربرگیرنده مقدار جغرافیائی است که به‌عنوان مختصات مرکز نقشه مورداستفاده قرار می‌گیرد.

map.getBounds(data, options?)

پارامترها:

- LngLat[]|string|Object data

مقدار بازگشتی: LatLngBounds

مقدار بازگشتی حاوی بازه‌ای از دو کرانه شمال شرق و جنوب غرب است که هر کران حاوی مختصات جغرافیائی مربوط به خود است. پارامتر `data` دربرگیرنده مجموعه‌ای از مختصات جغرافیائی است و از طریق تعیین این مختصات مقدار کران به دست خواهد آمد. با تعیین نوع منبع می‌توان نوع عارضه را بر اساس نوع‌های `tsv`، `geojson` مستقیماً جهت به دست آوردن کران‌ها به این تابع داد، ازاین‌رو با تعیین نوع و قرار دادن مقدار مربوط به نوع در `data` می‌توان مقدار را به دست آورد، کران‌ها در اشکال هندسی مانند خطوط و اشکال چندضلعی قابل دریافت است، برای به دست آوردن آن نیاز به حداقل دونقطه جغرافیایی است.

map.setBounds(bounds)

پارامترها:

- LngLatBounds bounds

مقدار بازگشتی: فاقد مقدار

کران نقشه را تعیین می‌کند به‌طوری‌که سطح بزرگ‌نمائی و مختصات مرکز نقشه محاسبه و تعیین می‌گردد. پارامتر **bounds** دربرگیرنده بازه‌ای از دو کرانه شمال شرق و جنوب غربی است که توسط این دونقطه مختصاتی سطح بزرگ‌نمائی و مختصات مرکز نقطه تنظیم می‌گردد.

map.panTo(center)

پارامترها:

- LngLat center

مقدار بازگشتی: فاقد مقدار

مختصات مرکز نقشه را مشابه **setCenter** تعیین می‌کند و هنگام تعیین مختصات مرکز نقشه از مکان فعلی خود به نقطه پیشنهادی به‌صورت متحرک جابه‌جا می‌گردد، پارامتر **center** دربرگیرنده مقدار جغرافیائی است که به‌عنوان مختصات مرکز نقشه مورداستفاده قرار می‌گیرد.

map.flyTo(center, zoom, speed)

پارامترها:

- LngLat center
- number zoom
- number speed

مقدار بازگشتی: فاقد مقدار

مختصات مرکز و سطح بزرگ‌نمائی به همراه میزان سرعت جابه‌جایی نقشه را با مقادیر موجود تعیین می‌کند. پارامتر **center** مربوطه به مختصات مرکز نقشه است. **zoom** سطح بزرگ‌نمائی نقشه را تعیین کرده و **speed** میزان سرعت جابه‌جایی و حرکت نقشه را به سمت نقطه پیشنهادی تعیین می‌کند به‌طوری‌که عدد **2** یعنی دو برابر حالت معمول و حالت معمول برابر عدد **1** است و این عدد می‌تواند به‌صورت اعشار نیز بکار گرفته شود.

map.zoomIn()

پارامترها: فاقد ورودی

مقدار بازگشتی: فاقد مقدار

سطح بزرگ‌نمائی نقشه یک سطح افزایش می‌یابد.

map.zoomOut()

پارامترها: فاقد ورودی

مقدار بازگشتی: فاقد مقدار

سطح بزرگ‌نمائی نقشه یک سطح کاهش می‌یابد.

map.remove()

پارامترها: فاقد ورودی

مقدار بازگشتی: فاقد مقدار

حذف کامل نقشه و بازگرداندن تگ `div` که دربرگیرنده `element` های نقشه است.

رخدادهای نقشه

رخدادهای نقشه در حین انجام عملی خاص بروی نقشه صدا زده می‌شود. برای استفاده از هر یک از eventها می‌بایست از طریق جایگیری در متد map.on به‌صورت map.on(eventName, handler) باشد که eventName همان نام یکی از eventهایی است که در این بخش قرار دارد. همچنین برای حذف یک رخداد می‌بایست از متد map.off استفاده نمود.

load

آرگومان‌ها:

- Map map

هنگام بارگذاری و ایجاد نقشه به‌صورت کامل، صدا زده می‌شود. ارگومان map ارائه‌دهنده شیء ای از نقشه است که قبلاً توسط کلاس Map ایجادشده است.

click

آرگومان‌ها:

- Object event
  - LngLat lngLat

هنگام کلیک بروی نقشه صدا زده می‌شود. ارگومان event دربرگیرنده ارگومان lngLat یا مختصات جغرافیایی نقطه کلیک شده بروی نقشه است.

mousedown

آرگومان‌ها:

- Object event
  - LngLat lngLat

هنگام کلیک بروی نقشه صدا زده می‌شود. ارگومان event دربرگیرنده ارگومان lngLat یا مختصات جغرافیایی نقطه کلیک شده بروی نقشه است. این رخداد بلافاصله پس از کلیک بروی نقشه صدا زده شده و هنگام جابه‌جایی نقشه نیز قبل از رخدادهای دیگر نقشه اجرا می‌شود.

mousemove
آرگومان‌ها:

- `Object event`
  - `LngLat lngLat`

هنگام جابه‌جایی اشاره‌گر ماوس بروی نقشه صدا زده می‌شود. ارگومان `event` دربرگیرنده ارگومان `lngLat` یا مختصات جغرافیایی نقطه‌ای از نقشه است که ماوس در آن قرار دارد.

movestart
آرگومان‌ها: فاقد ورودی

بلافاصله پس‌ازاینکه جابه‌جایی نقشه شروع گردد، صدا زده می‌شود.

moveend
آرگومان‌ها: فاقد ورودی

پس‌ازاینکه جابه‌جایی نقشه خاتمه یافت، صدا زده می‌شود.

zoomstart
آرگومان‌ها: فاقد ورودی

بلافاصله پس‌ازاینکه تغییر سطح بزرگ‌نمائی نقشه شروع گردد، صدا زده می‌شود.

zoomend
آرگومان‌ها: فاقد ورودی

پس‌ازاینکه تغییر سطح بزرگ‌نمائی نقشه خاتمه یافت، صدا زده می‌شود.

styleload
آرگومان‌ها: فاقد ورودی

هنگام تغییر استایل نقشه صدا زده می‌شود. به هنگام تغییر استایل نقشه تمامی عارضه‌های موجود از نقشه حذف و کل نقشه دوباره از نو بروی `canvas` ترسیم می‌شود ازاین‌رو این رخداد هنگامی‌که نقشه به‌طور کامل ترسیم شد، صدا زده می‌شود و می‌توان عارضه‌های قبلی را پس از بارگذاری مجدد استایل به نقشه افزود.

منابع داده‌های جغرافیائی

منظور از source مجموعه داده‌های جغرافیائی هستند که دربرگیرنده عارضه‌های روی نقشه است. این منابع می‌توانند شامل feature های مربوط به Point، LineString و Polygon باشند. در حال حاضر چند نوع ورودی می‌توانند به‌عنوان منابع داده‌های جغرافیایی در نظر گرفته شوند.

نوع داده‌های جغرافیائی

GoeJson

یک نوع رایج از داده‌های جغرافیایی است که از مجموعه‌ای از عارضه‌ها یا FeatureCollection تشکیل‌شده است.

TSV

فرمت TSV مخفف عبارت Tab Separated Value است، بدین معنا که توسط جداولی از سطرها و ستون‌های تشکیل‌شده که سطرها و ستون‌ها به ترتیب از طریق newline و واژه tab از هم جداشده‌اند. اولین سطر این جداول مربوط به header یا کلیدهاست که چینش مقادیر در سطرهای بعدازآن بر اساس header صورت می‌پذیرد. فیلدهای داده‌های جغرافیائی نیز از نوع WKT یا Well Know Text است. داده‌های TSV برخلاف GeoJson است، بدین معنی که از JSON Syntax استفاده نکرده و قالب قابل‌قبول برای آن رشته است به‌طوری‌که اغلب درون فایل‌های متنی قرار می‌گیرند.

افزودن منبع

جهت افزودن یک منبع از متد map.addSource استفاده می‌شود. داده‌ها توسط فرمت‌های شرح داده‌شده به نقشه جهت استفاده افزوده می‌شوند و فرمت پیش‌فرض برای افزودن داده برابر geojson است.

```
map.addSource('point', {
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "type": "Point",
        "coordinates": [51.41, 35.7575]
      }
    }
```

```
  ]
});
```

در نمونه فوق یک منبع با شناسه **point** به نقشه افزوده می‌شود داده استفاده‌شده در این منبع از نوع **GeoJson** بوده و شامل یک عارضه **Point** است که از آن می‌توان برای افزودن یک‌لایه **symbol** یا **circle** به نقشه استفاده نمود.

```
var tsvData = `wkt
POINT (51.40995651483536 35.75752664431239)`;

map.addSource('point', tsvData, { type: 'tsv' });
```

در این نمونه کد مشابه مثال قبلی یک منبع با شناسه **point** به نقشه افزوده می‌شود که داده استفاده‌شده در این منبع از نوع **TSV** بوده و شامل یک عارضه **Point** است.

```
map.addSource("data", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { title: "My Marker", value: 20 },
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7575]
      }
    },
    {
      type: "Feature",
      properties: { tile: "My line", value: 30 },
      geometry: {
        type: "LineString",
        coordinates: [[51.41, 35.7575], [51.4, 35.7575], [51.4, 35.75]]
      }
    }
  ]
});
```

در نمونه کد دو عارضه **Point** و **LineString** به منبع افزوده می‌شوند این مجموعه داده‌ها می‌توانند به ترتیب در لایه‌هایی از نوع **symbol** و **circle** برای **Point**و لایه **line** برای **LineString** به کار گرفته شوند. همچنین از مقادیر موجود در **properties** نیز می‌توان در روند ساخت لایه به‌عنوان داده پویا بهره جست.

```
var tsvData = `"wkt title   value
POINT (51.41 35.7575)   My Marker   20
LINESTRING (51.41 35.7575,51.4 35.7575,51.4 35.75)      30"`;

map.addSource('data', tsvData, { type: 'tsv' });
```

در این نمونه کد مشابه مثال قبل داده‌های جغرافیایی **Point** و **LineString** به منبع افزوده می‌شوند و همان‌طور که در بخش **TSV** شرح داده شد مقادیر توسط کاراکتر **tab** از هم جدا می‌گردند و اولین سطر داده شامل **header** جدول اطلاعاتی است.

دریافت داده‌های جغرافیائی از منبع

جهت دریافت داده‌های افزوده‌شده به نقشه از متد map.getSource استفاده نمود. این داده‌ها توسط شناسه در دسترس قابل دریافت بوده و همچنین می‌تواند شکل بازگشتی در قالب فرمت‌های قابل‌ارائه باشد. فرمت پیش‌فرض برای دریافت داده برابر **geojson** است.

```
var source = map.getSource('point');
var geoJson = source.data;
```

در نمونه کد مجموعه‌ای از داده‌های جغرافیائی قابل دریافت است. داده‌های درون منبع **point** متشکل از عارضه‌هایی است که توسط فرمت مربوطه به نقشه افزوده‌شده است. این تابع دارای دو مقدار **data** و **setData** است که در اینجا از **data** استفاده‌شده است، **data** شامل داده‌های مربوط به منبع **point** است.

```
var source = map.getSource('point', { type: 'tsv' });
var tsvData = source.data;
```

در این نمونه کد **data** از مجموعه‌ای از دادهایی تشکیل می‌شود که قبلاً به نقشه افزوده‌شده‌اند و شکل خروجی تابع که درون مقدار **data** قرار دارد برابر **tsv** یا ساختار جدولی شکل است.

تغییر داده‌های جغرافیائی یک منبع

برخی اوقات لازم است که در حین نمایش عارضه‌ها توسط منابع و ایجاد لایه، مقادیر داده‌ها تغییر یابند ازاین‌رو برای تغییر داده‌های موجود در یک منبع داده‌ای از متد **setData** استفاده می‌شود این متد توسط متد map.getSource بکار گرفته می‌شود. اختصاص داده‌های جدید به یک منبع سبب می‌گردد تمامی داده‌های قبلی با داده‌های جدید جایگزین گردد، ازاین‌رو توجه به این نکته مهم است که قبل از اختصاص داده‌های جدید، بررسی شود که چه دیتایی می‌بایست به‌عنوان داده‌های بروز شده درون یک منبع جای گیرد. فرمت پیش‌فرض برای دریافت داده برابر **geojson** است.

```
var geoJson = {
```

```
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { title: "My Marker", value: 50 },
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7576]
      }
    }
  ]
};
var source = map.getSource("point");
var oldGeoJson = source.data;
source.setData(geoJson);
```

در نمونه کد توسط متد map.getSource منبعی از نقشه قابل دریافت است. داده‌های این منبع می‌تواند بعداً در طول روند برنامه تغییریافته و دوباره از طریق متد **setData** که از منبع قابل‌استفاده است به منبع مدنظر بازگردد. در این نمونه داده‌های قبلی در متغیر **oldGeoJson** قرار یافته‌اند و داده‌های جدیدی که درون **geoJson** قرار دارند به منبع اختصاص داده می‌شود.

```
var tsvData = `wkt
POINT (51.40995651483536 35.75752664431239)`;

var source = map.getSource("point");
map.setSource(tsvData, { type: "tsv" });
```

داده‌ها را می‌توان توسط فرمت‌های رایج نیز به منبع اختصاص داد، در نمونه کد داده در فرمت **tsv** در منبع **point** قرار یافته است.

حذف یک منبع

جهت حذف یک منبع با تمامی داده‌ها از متد map.removeSource استفاده می‌شود. در این متد نیاز به اختصاص شناسه منبع مدنظر برای حذف آن از نقشه، خواهد بود.

لایه

لایه‌ها بروی نقشه جهت نمایش عارضه‌های مختلف ازجمله **symbol**، **line**، **fill**، **heatmap** و **circle** کاربرد دارند و در کل تعامل با عارضه‌ها را فراهم کرده و می‌توان از طریق آن نحوه و سبک نمایش عارضه‌ها را نیز بروی نقشه تعیین نمود. هر لایه به‌وسیله داده‌های منابع قادر به نمایش آن بروی نقشه است و منابع تنها مسئول اختصاص دادن داده برای استفاده در لایه هستند.

رخدادهای لایه

رخدادهای لایه در حین انجام عملی خاص بروی لایه صدا زده می‌شود. برای استفاده از هر یک از **event**ها می‌بایست از طریق جایگیری در تابع `map.on` به‌صورت `map.on(eventName, layerId, layerHandler)` باشد که **eventName** همان نام یکی از **event**هایی است که در این بخش قرار دارد و منظور از **layerId** شناسه لایه‌ای است که باید رخداد به آن افزوده شود. همچنین برای حذف یک رخداد می‌بایست از تابع `map.off` استفاده نمود.

click

آرگومان‌ها:

- Object event
  - LngLat lngLat
  - number id

هنگام کلیک بروی یک‌لایه صدا زده می‌شود. ارگومان **event** دربرگیرنده ارگومان **lngLat** یا مختصات جغرافیایی است و درصورتی‌که لایه از نوع **symbol** باشد مقدار **lngLat** برابر به مختصات **Point** موردنظر است ولی درصورتی‌که نوع دیگری باشد نقطه کلیک شده بروی لایه به‌عنوان **lngLat** در دسترس قرار خواهد گرفت، ارگومان **id** مربوط به شناسه‌ای است که هنگام ایجاد منبع برای داده در نظر گرفته‌شده و فیلد **id** یکی از داده‌های پویائی است که درون **properties** مربوط به عارضه مدنظر قرار می‌گیرد.

گزینه‌های لایه

هر لایه شامل **5** گزینه است که هریکی از این موارد وظیفه خاصی را بر عهده‌دارند در زیر به برخی از ویژگی‌های هر گزینه پرداخته می‌شود. لازم به ذکر است که در هر بخش از تعریف لایه‌ها تمامی خصوصیت‌های مربوط به لایه به‌طور کامل شرح داده‌شده و در اینجا تنها به برخی از خصوصیت‌های مربوط به هر گزینه قرار دارد.

- **layout**
این گزینه می‌تواند خصوصیت‌های خاصی را برای کنترل بروی موقعیت آیکون در لایه **symbol** یا افزودن **popup** به عارضه‌ها و یا پنهان‌سازی یا نمایش یک‌لایه را بر عهده گیرد.
- **paint**
این گزینه کلیه اعمال گرافیکی بروی لایه را بر عهده دارد، برای نمونه می‌توان رنگ متن، رنگ مربوط به پس‌زمینه اشکال هندسی و خطوط و سایر موارد گرافیکی را در **paint** تعیین نمود.
- **filter**
تمامی شروط خاص برای اعمال سایر گزینه‌ها بروی یک‌لایه را بر عهده دارد در این گزینه عملگرهای منطقی سبب می‌شوند تا درصورتی‌که مقدار شرط بر قرار بود لایه در نقشه نمایش یابد.
- **minzoom**
توسط این گزینه حداقل سطح بزرگ‌نمائی که لایه در آن قادر به نمایش بروی نقشه است، تعیین می‌گردد.
- **maxzoom**
توسط این گزینه حداکثر سطح بزرگ‌نمائی که لایه در آن قادر به نمایش بروی نقشه است، تعیین می‌گردد.

تغییر خصوصیت‌های layout

توسط متد map.setLayerLayout می‌توان خصوصیت‌های **layout** مربوط به یک‌لایه را تغییر داد، این تغییرات سبب جایگزین شدن با خصوصیات قبلی می‌گردد. در نمونه کد خصوصیت مربوط به نمایش لایه مجدداً با مقدار جدید تعیین می‌گردد.

```
map.setLayerLayout("line", { "visibility": "visible" });
```

تغییر خصوصیت‌های paint

توسط متد map.setLayerPaint می‌توان خصوصیت‌های **paint** مربوط به یک‌لایه را تغییر داد، این تغییرات سبب جایگزین شدن با خصوصیات قبلی می‌گردد. در نمونه کد خصوصیت مربوط به تغییر رنگ پس‌زمینه مربوط به عارضه چندضلعی در لایه **fill** مجدداً با مقادیر جدید تعیین می‌گردد.

```
map.setLayerPaint("fill", { "fill-color": "#126" });
```

تغییر خصوصیت filter

توسط متد map.setLayerFilter می‌توان خصوصیت filter مربوط به یک‌لایه را تغییر داد، این تغییرات سبب جایگزین شدن با خصوصیات قبلی می‌گردد. در نمونه کد خصوصیت مربوط به تعیین شرط با مقادیر جدید تعیین می‌گردد. در این نمونه عارضه‌ای که در فیلد value که به‌صورت داده پویا است و درون properties تعیین‌شده، حاوی مقداری بزرگ‌تر از 50 باشد، گزینه‌های لایه بروی آن اعمال خواهد شد.

```
map.setLayerFilter("symbol", ["<=", ["get", "value"], 50]);
```

دریافت اطلاعات لایه

جهت دریافت اطلاعات مربوط به یک‌لایه لازم است که شناسه مربوط به لایه مدنظر را در متد map.getLayer اختصاص داد. مقدار دریافتی می‌تواند دربرگیرنده اطلاعاتی باشد که قبلاً به لایه اختصاص داده‌شده است. ازاین‌رو این متد نیز می‌تواند یک‌لایه را از جهت وجود داشتن آن بررسی کند.

حذف لایه

جهت حذف یک‌لایه از نقشه از متد map.removeLayer استفاده می‌شود، در این متد نیاز به اختصاص شناسه لایه مدنظر برای حذف آن از نقشه، خواهد بود.

لایه symbol

لایه symbol قادر است جزئیات گرافیکی مربوط به نقاط را بروی نقشه نمایش دهد، داده‌هایی که از نوع Point به منبع اختصاص‌یافته‌اند برای این نوع لایه مورداستفاده قرار می‌گیرد. symbol برای نمایش یک marker، نمادی از مکان و تصاویر بندانگشتی بروی نقشه برای نمایش یک موقعیت خاص کاربرد دارند. برای اینکه یک‌لایه symbol را به نقشه افزود در ابتدا نیاز به اماده‌سازی تصاویری مربوط به هر لایه است.

اماده‌سازی تصاویر

قبل از به‌کارگیری لایه symbol نیاز به بارگذاری و اماده‌سازی کلیه تصاویر مورداستفاده در نقشه است. توسط متد map.loadImageCollection می‌توان مجموعه تصاویری که لازم است توسط لایه symbol بروی نقشه قرار یابند، اختصاص داد این تابع علاوه بر دریافت مجموعه تصاویر از نوع ImageCollection، یک تابع را نیز به‌صورت callback در ورودی دریافت می‌کند که به‌وسیله تعیین این تابع هنگام بارگذاری کامل تصاویر بروی نقشه این متد صدا زده شده و ازاین‌رو ایجاد مؤثر لایه‌ها امکان‌پذیر خواهد شد، به‌طوری‌که نیاز است افزودن لایه توسط متد map.addSymbolLayer درون این تابع قرار بگیرد. عدم قرارگیری متد افزودن لایه symbol به‌عنوان callback درون این تابع سبب می‌گردد که هنگام افزوده شدن لایه‌ها عارضه‌ها فاقد تصویر بوده و هیچ تصویری بروی نقشه مشاهده نگردد. لزوم استفاده ازاین‌روش بدین دلیل است که در این نسخه بجای استفاده از المنت‌های تکی به‌عنوان عارضه بروی نقشه از یک بوم یا canvas برای render نمودن تمامی عارضه‌ها بروی نقشه با سرعت و کیفیت بالا استفاده‌شده است، بااین‌حال برای استفاده مؤثر از لایه‌های symbol نیز نیاز است تا تصاویر برای قرارگیری روی بوم آماده و توسط شروط موجود در لایه در زمانی خاص مشاهده شود. متد map.loadImageCollection حامل داده‌های مربوط به تصاویر است که اولین پارامتر آن شامل مجموعه‌ای از داده‌هاست که هر یک از اعضای این مجموعه از دو بخش name و url تشکیل‌شده است که name به‌عنوان نام تصویری است که در روال افزودن لایه از آن استفاده می‌شود و url نیز مربوط به آدرس فیزیکی تصویر است که باید قبل از افزودن به تصویر از این آدرس خوانده شود.

افزودن لایه symbol

برای افزودن یک‌لایه symbol همان‌طور که در بخش قبل بیان گردید، نیاز به استفاده از متد map.loadImageCollection است توسط این متد کلیه تصاویر مورداستفاده در لایه به نقشه جهت استفاده افزوده می‌شود. توسط متد map.addSymbolLayer یک‌لایه از نوع symbol به نقشه افزوده می‌شود، هر لایه symbol حامل یک شناسه لایه و یک شناسه منبع است ازاین‌رو می‌بایست ابتدا مطابق

بخش افزودن منبع منبعی خاص به نقشه افزوده شود و توسط اختصاص شناسه منبع به لایه از داده‌های آن به‌صورت نمایش بروی نقشه بهره جست.

```
map.addSource("point", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7575]
      }
    }
  ]
});
var imageCollection = [{ name: "drone", url: "HOST_URL/images/drone.png" }];
map.loadImageCollection(imageCollection, function() {
  map.addSymbolLayer("marker", "point", { layout: { "icon-image": "drone" } });
});
```

در نمونه بالا ابتدا منبعی به نقشه با شناسه **point** افزوده‌شده و حاوی داده‌ای از نوع **Point** است، در گام بعد متغیر **imageCollection** حاوی داده‌های تصویری که در این مثال مورداستفاده قرار می‌گیرد، این داده‌ها حاوی نام **drone** و یک آدرس تصویری مربوط به آن است. توسط متد map.loadImageCollection نیز این داده‌ها بارگذاری می‌شوند و پس از بارگذاری تابع **callback** فراخوانی می‌شود درون این تابع از متد map.addSymbolLayer استفاده‌شده است به‌طوری‌که این متد با شناسه **marker** و با شناسه منبع **point** و با گزینه‌های لایه ازجمله **layout** به نقشه افزوده‌شده است. درون گزینه‌های تابع از خصوصیت **icon-image** استفاده‌شده است این خصوصیت قادر است که یکی از تصاویر موجود در نقشه را برای لایه بر اساس **name** که در **imageCollection** مورداستفاده قرارگرفته، در نظر بگیرد.

استفاده از داده‌های پویا

در بخش افزودن منبع شرح داده شد که می‌توان از خصوصیت‌های **properties** درون لایه‌ها استفاده نمود. داده‌های تغییرپذیر که درون **properties** قرار می‌یابند می‌توانند دربرگیرنده نام تصاویری باشند که لازم است هنگام افزودن لایه از آن استفاده شود. برای نمونه **imageCollection** دربرگیرنده دو تصویر بانام‌های **image1** و **image2** است در این صورت درون داده‌های موجود در **properties** می‌توان در یک

فیلد مشخص مانند **image** این تصاویر را برای هر عارضه قرار داد. بدین شکل هنگام افزودن لایه و با تعیین داده‌های پویا می‌توان تصاویر را بر اساس فیلد مدنظر به‌صورت **dynamic** در نقشه نمایش داد.

```
map.addSource("point", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { image: "drone" },
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7575]
      }
    },
    {
      type: "Feature",
      properties: { image: "solar-panel" },
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7576]
      }
    }
  ]
});
var imageCollection = [
  { name: "drone", url: " HOST_URL/images/drone.png" },
  { name: "solar-panel", url: " HOST_URL/images/solar-panel.png" }
];
map.loadImageCollection(imageCollection, onImageCollectionLoad);
function onImageCollectionLoad() {
  map.addSymbolLayer("marker", "point", {
    layout: { "icon-image": "{image}" }
  });
}
```

در نمونه یک‌لایه با داده‌های پویا به نقشه افزوده می‌شود داده‌های بکار گرفته‌شده در **properties** با توجه به دو عارضه مشخص‌شده حاوی یک فیلد اطلاعاتی **image** است که این فیلدها به ترتیب شامل **drone** و **solar-panel** است و در اصل همان نام تصاویری است که هنگام بارگذاری تصاویر تعیین‌شده‌اند. درون متغیر **imageCollection** مجموعه‌ای از تصاویر مورداستفاده در نقشه اختصاص

می‌یابند که شامل دو تصویر بانام‌های **drone** و **solar-panel** است، درون متد **callback** آن، تابع **onImageCollectionLoad** قرار یافته که این تابع دربرگیرنده متد map.addSymbolLayer است. درون **layerOptions** این متد از گزینه **layout** و درنهایت یکی از خصوصیت‌های آن به نام **icon-image** استفاده‌شده است این خصوصیت حاوی مقدار **{image}** است که این نوع استفاده بدین معنی است که از داده‌های پویایی که در بخش **properties** مربوط به منبع داده جغرافیائی اختصاص‌یافته، استفاده نموده و فیلدهای اطلاعاتی می‌توانند با اسامی دلخواه هم مورداستفاده قرار گیرند.

گزینه‌های لایه symbol

هر لایه **symbol** می‌تواند شامل گزینه‌های مختلفی ازجمله **layout**، **paint**، **filter**، **minzoom** و **maxzoom** باشد. هدف استفاده از گزینه‌های لایه این است که با استفاده از آن‌ها **marker** خاص بروی نقشه با توجه به تعیین تصویر، در جهتی خاص بعد از افزودن لایه یا بعد از تعیین **visibility** نمایش یابد. همچنین در **layout** می‌توان **popup** و متنی خاص را بروی نقشه به نمایش در آورد و در **paint** نیز می‌توان رنگ متن مدنظر را تعیین نموده و درنهایت در گزینه **filter** می‌توان از شروط خاصی برای نمایش لایه استفاده نمود. از گزینه‌های **minzoom** و **maxzoom** به ترتیب برای تعیین محدوده نمایش عارضه بروی نقشه استفاده می‌گردد به‌طوری‌که بر اساس تعیین محدوده سطح بزرگ‌نمائی در **zoom level** مدنظر عارضه به نمایش در خواهد آمد.

خصوصیت‌های layout

- **icon-image**

اختصاص نام یک تصویر به‌صورت رشته یا استفاده از داده‌های پویا، در نمونه کد زیر با استفاده از فیلد **image** از **properties** مربوط به منبع تصویر به عارضه اختصاص‌یافته است.

```
map.addSymbolLayer("marker", "point", {
  layout: {"icon-image": "{image}"}
});
```

- **icon-anchor**

اختصاص موقعیت آیکون در بالا، پایین یا مرکز مختصات، در حالت‌هایی خاص ممکن است نیاز باشد که آیکون با توجه به شکل در مرکز مختصات قرار گیرد ولی در حالت معمول اگر آیکون به‌صورت **marker** دارای یک ضلع رو به پایین باشد قرارگیری آیکون بهتر است در جهت رو به پایین نقطه باشد ازاین‌رو ضلع پایین تصویر بروی مختصات قرار می‌گیرد ولی در اشکال دیگر که به‌صورت مارکر نبوده و یا یک شکل شبیه به مستطیل، دایره و سایر اشکال هندسی دارند می‌توان

آن را در مرکز مختصات جغرافیایی قرار دارد. رایج‌ترین نوع‌ها شامل **top**، **center** و **bottom** است. به‌طور پیش‌فرض این مقدار برابر **center** یا مرکز مختصات جغرافیائی است. در نمونه کد زیر با استفاده از فیلد **properties** مربوط به منبع، تصویری به عارضه اختصاص‌یافته است و همچنین موقعیت قرارگیری آیکون در ضلع جنوبی آن بروی مختصات قرار یافته است.

```
map.addSymbolLayer("marker", "point", {
  layout: { "icon-image": "{image}", "icon-anchor": "bottom" }
});
```

- **visibility**

این خصوصیت سبب نمایش لایه بروی نقشه می‌شود، گزینه‌های موجود در این خصوصیت شامل **visible** و **none** است و مقدار پیش‌فرض آن برابر **visible** است، به‌طوری‌که هنگام افزودن یک‌لایه بلافاصله عارضه نمایان می‌گردد می‌توان با استفاده از مقدار **none** لایه را افزوده ولی نمایش نداد سپس توسط متد map.setLayerLayout خصوصیت **visiblity** را به مقدار دلخواه تغییر داد. در نمونه زیر لایه ایجاد می‌شود ولی پس از افزوده شدن نمایش نمی‌یابد.

```
map.addSymbolLayer("marker", "point", {
  layout: { "icon-image": "{image}", visibility: "none" }
});
```

- **popup-text**

با افزودن این خصوصیت یک کادر که معمولاً حاوی اطلاعاتی در مورد عارضه مدنظر است بروی آن ظاهر می‌شود. که با کلیک بروی عارضه این کادر اطلاعاتی بروی آن ظاهر می‌شود. در نمونه زیر متن مدنظر با کلیک بروی عارضه بروی آن ظاهر می‌شود. این خصوصیت می‌تواند همچنین از داده‌های پویای تعیین‌شده در **properties** منبع نیز استفاده کند. در این صورت شکل ورودی با توجه به نمونه دوم تغییر می‌یابد. که در آن از فیلد **text** استفاده‌شده است.

```
map.addSymbolLayer("marker", "point", {
  layout: { "icon-image": "{image}", "popup-text": "The popup text." }
});
map.addSymbolLayer("marker", "point", {
  layout: { "icon-image": "{image}", "popup-text": "{text}" }
});
```

- **popup-type**

این خصوصیت سبب می‌گردد تا نوع ظاهر شدن کادر اطلاعاتی را تعیین کند. مقدار پیش‌فرض آن برابر **click** است و دو مقدار **click** و **hover** را می‌تواند در بر گیرد. در نوع **click** با کلیک

بروی عارضه **popup** ظاهر می‌شود و با تعیین نوع **hover** کادر اطلاعاتی به محض قرارگیری ماوس بروی آیکون، ظاهر می‌گردد. در نمونه زیر از نوع **hover** استفاده‌شده است و برای به‌کارگیری آن لازم است که از خصوصیت **popup-text** هم استفاده‌شده باشد.

```
map.addSymbolLayer("marker ", "point", {
  layout: {
    "icon-image": "{image}",
    "popup-text": "{text}",
    "popup-type": "hover"
  }
});
```

- **popup-offset**

زمانی که از خصوصیت **icon-anchor** برای تعیین موقعیت آیکون بروی مختصات جغرافیایی مدنظر استفاده می‌شود، لازم است که جایگاه دقیق آن را برای ظاهر شدن **popup** نیز تعیین نمود. علاوه بر این خصوصیت **popup-offset** تعیین‌کننده موقعیت ظاهر شدن کادر اطلاعاتی بروی آیکون است. و این خصوصیت بر اساس تناسب آن با اندازه آیکون باید با مقادیر مناسب پر گردد مقدار باید به‌صورت آرایه‌ای از دو عدد باشد که نشان‌گر مقادیر مربوط به **x** و **y** فاصله از نقطه مدنظر است. در نمونه زیر **popup** در نقاط **20** و **30** پیکسل فاصله از مرکز یا مختصات جغرافیائی بروی آیکون نمایش می‌یابد.

```
map.addSymbolLayer("marker ", " point ", {
  layout: {
    "icon-image": "{image}",
    "popup-text": "{text}",
    "popup-offset": [20, 30]
  }
});
```

- **text-field**

این خصوصیت سبب می‌گردد تا لایه مدنظر حاوی یک فیلد متنی باشد به‌طوری‌که در نقطه جغرافیائی خاص متن موردنظر قرار می‌گیرد. در نمونه زیر متن مدنظر در مختصات مربوطه بروی نقشه جای می‌گیرد. این فیلد می‌تواند دربرگیرنده داده‌های پویا باشد به‌طوری‌که می‌توان متن را از درون **properties** مربوط به منبع مدنظر دریافت و به نمایش بگذارد، در نمونه دوم نمایش متن از طریق داده پویا امکان‌پذیر شده است. این خصوصیت برای نمایش متن بروی نقشه نیازی به **icon-image** نداشته و اگر هم از خصوصیت **icon-image** استفاده‌شده باشد متن بر روی آیکون قرار می‌گیرد.

```
map.addSymbolLayer("marker", "point", {
  layout: { "text-field": "The text" }
});
map.addSymbolLayer("marker", "point", {
  layout: { "text-field": "{text}" }
});
```

خصوصیت‌های paint

- text-color

این خصوصیت رنگی را با فیلدهای متنی اختصاص می‌دهد، لزوم اعمال این کار وجود داشتن یک فیلد متنی در خصوصیت‌های layout است. در نمونه کد زیر متن با کد رنگ #d85471 بروی نقشه نمایش می‌یابد.

```
map.addSymbolLayer("marker", "point", {
  layout: { "text-field": "{text}" },
  paint: { "text-color": "#d85471" }
});
```

خصوصیت filter

توسط فیلتر می‌توان نمایش عارضه‌ها بروی نقشه را بر اساس قاعده و شروطی خاص تعیین نمود. در شروط می‌توان از عبارات منطقی مانند >، <، => و =< نیز بهره برد. برای نمونه در کد زیر زمانی که مقدار image درون properties مربوط به منبع داده جغرافیایی برابر مقدار solar-panel باشد لایه تنها بروی آن داده جغرافیائی قابل‌اجرا است.

```
map.addSymbolLayer("marker", "point", {
  layout: { "icon-image": "{image}" },
  filter: ["==", ["get", "image"], "solar-panel"]
});
```

خصوصیت‌های zoom

هر لایه را می‌توان در محدوده‌ای از سطوح بزرگ‌نمائی نمایش داد که برای این کار از خصوصیت‌های minzoom و maxzoom استفاده می‌شود. در نمونه زیر لایه مدنظر تنها در بین دو سطح بزرگ‌نمائی 10 تا 18 قابل روئیت بروی نقشه است. همچنین می‌توان خصوصیت‌های مربوط به سطوح بزرگ‌نمائی را به‌صورت تکی مورداستفاده قرار داد برای نمونه با تعیین minzoom حداقل سطح بزرگ‌نمائی و با تعیین maxzoom حداکثر سطح بزرگ‌نمائی را برای روئیت عارضه بروی نقشه تعیین نمود.

```
map.addSymbolLayer("marker", "point", {
```

```
  layout: { "icon-image": "{image}" },
  minzoom: 10,
  maxzoom: 18
});
```

دریافت اطلاعات symbol پیش‌فرض

توسط متد map.getDefaultSymbol می‌توان به آیکون مارکر پیش‌فرض به همراه گزینه‌های لایه دسترسی داشت. آیکون مارکر پیش‌فرض می‌تواند ارائه‌دهنده یک آیکون باشد ازاین‌رو می‌تواند به تنهائی درون گزینه‌های لایه استفاده شود.

```
var defaultSymbol = map.getDefaultSymbol();
map.addSymbolLayer("marker", "point", defaultSymbol);
```

در نمونه کد لایه‌ای با نام **marker** و با استفاده از منبعی با شناسه **point** ایجاد می‌شود. گزینه‌های لایه ایجادشده بر اساس مارکر پیش‌فرض در نظر گرفته می‌شود که مقدار آن درون **defaultSymbol** قرار دارد.

لایه line

لایه **line** قادر است جزئیات گرافیکی مربوط به خطوط را بروی نقشه نمایش دهد، داده‌هایی که از نوع **lineString** به منبع اختصاص‌یافته‌اند برای این نوع لایه مورداستفاده قرار می‌گیرد. **line** برای نمایش یک پاره‌خط یا مجموعه‌ای خطوط کاربرد دارند.

افزودن لایه line

توسط متد map.addLineLayer یک‌لایه از نوع **line** به نقشه افزوده می‌شود، هر لایه **line** حامل یک شناسه لایه و یک شناسه منبع است ازاین‌رو می‌بایست ابتدا مطابق بخش افزودن منبع منبعی خاص به نقشه افزوده شود و توسط اختصاص شناسه منبع به لایه از داده‌های آن به‌صورت نمایش بروی نقشه بهره جست.

```
map.addSource("linestring", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { value: 30 },
      geometry: {
        type: "LineString",
        coordinates: [[51.41, 35.7575], [51.4, 35.7575], [51.4, 35.75]]
      }
    }
  ]
});
map.addLineLayer("line", "linestring");
```

در نمونه کد ابتدا منبعی به نقشه با شناسه **linestring** افزوده می‌شود و حاوی داده‌ای از نوع **LineString** است. با استفاده از متد map.addLineLayer لایه‌ای از نوع **line** با شناسه **line** و همچنین با استفاده از **linestring** به نقشه افزوده می‌شود. به‌طور پیش‌فرض خصوصیت‌های مربوط به **paint** شامل **line-color** برابر رنگ مشکی و **line-width** برابر **1** پیکسل پهنا است.

استفاده از داده‌های پویا

در بخش افزودن منبع شرح داده شد که می‌توان از خصوصیت‌های **properties** درون لایه‌ها استفاده نمود. داده‌های تغییرپذیر که درون **properties** قرار می‌یابند می‌توانند دربرگیرنده مقادیری جهت کنترل **dynamic** بروی خطوط ترسیم‌شده باشد.

```
map.addSource("linestring", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { width: 5, color: "#874" },
      geometry: {
        type: "LineString",
        coordinates: [[51.41, 35.7575], [51.4, 35.7575], [51.4, 35.75]]
      }
    }
  ]
});
map.addLineLayer("line", "linestring", {
  paint: {
    "line-width": ["get", "width"],
    "line-color": ["get", "color"]
  }
});
```

در نمونه بالا یک‌لایه با داده‌های پویا به نقشه افزوده می‌شود داده‌های بکار گرفته‌شده در **properties** با توجه به عارضه خطی مشخص‌شده حاوی دو فیلد اطلاعاتی **width** و **color** است که این فیلدها را می‌توان به‌عنوان مقادیری برای خصوصیت‌های موجود مورداستفاده قرار داد. با توجه به مقادیر، پهنا و رنگ خط ترسیم‌شده بروی خطوط برابر مقادیر داده‌شده خواهد شد که این نوع استفاده بدین معنی است که از داده‌های پویایی که در بخش **properties** مربوط به منبع داده جغرافیائی اختصاص‌یافته، استفاده نموده و فیلدهای اطلاعاتی می‌توانند با اسامی دلخواه هم مورداستفاده قرار گیرند.

گزینه‌های لایه line

هر لایه **line** می‌تواند شامل گزینه‌های مختلفی ازجمله **layout**، **paint**، **filter**، **minzoom** و **maxzoom** باشد. هدف استفاده از گزینه‌های لایه این است که با استفاده از آن‌ها پاره‌خط یا خطوطی خاص بروی نقشه با توجه به تعیین پهنا، رنگ، سطح شفافیت و نوع الحاق با خطوط دیگر بعد از افزودن لایه یا بعد

از تعیین **visibility** نمایش یابد. همچنین در **layout** می‌توان **popup** و نوع الحاق خطوط را تعیین نمود و در **paint** نیز می‌توان رنگ و پهنای خط مدنظر را تعیین نموده و درنهایت در گزینه **filter** می‌توان از شروط خاصی برای نمایش لایه استفاده نمود. از گزینه‌های **minzoom** و **maxzoom** به ترتیب برای تعیین محدوده نمایش عارضه بروی نقشه استفاده می‌گردد به‌طوری‌که بر اساس تعیین محدوده سطح بزرگ‌نمائی در **zoom level** مدنظر عارضه به نمایش در خواهد آمد.

خصوصیت‌های layout

- **line-join**

نحوه و سبک الحاق خطوط به همدیگر را تعیین می‌کند. انواع مختلف قابل‌استفاده برای نوع الحاق برابر **miter** و **round** است که **miter** سبب می‌شود که نحوه اتصال خطوط به همدیگر دارای لبه باشد ولی در **round** اتصال به‌صورت منحنی صورت می‌پذیرد. مقدار پیش‌فرض آن برابر **miter** است. در نمونه زیر نحوه الحاق خطوط به‌صورت منحنی خواهد بود.

```
map.addLineLayer("line", "linestring", {
  layout: { "line-join": "round" }
});
```

- **visibility**

این خصوصیت سبب نمایش لایه بروی نقشه می‌شود، گزینه‌های موجود در این خصوصیت شامل **visible** و **none** است و مقدار پیش‌فرض آن برابر **visible** است، به‌طوری‌که هنگام افزودن یک‌لایه بلافاصله عارضه نمایان می‌گردد می‌توان با استفاده از مقدار **none** لایه را افزوده ولی نمایش ند سپس توسط متد map.setLayerLayout خصوصیت **visiblity** را به مقدار دلخواه تغییر داد. در نمونه زیر لایه ایجاد ولی پس از افزوده شدن نمایش نمی‌یابد.

```
map.addLineLayer("line", "linestring", { layout: { visibility: "none" } });
```

- **popup-text**

با افزودن این خصوصیت یک کادر که معمولاً حاوی اطلاعاتی در مورد عارضه مدنظر است بروی آن ظاهر می‌شود. که با کلیک بروی عارضه‌این کادر اطلاعاتی بروی آن ظاهر می‌شود. در نمونه زیر متن مدنظر با کلیک بروی عارضه بروی آن ظاهر می‌شود. این خصوصیت می‌تواند همچنین از داده‌های پویای تعیین‌شده در **properties** منبع نیز استفاده کند. در این صورت شکل ورودی با توجه به نمونه دوم تغییر می‌یابد. که در آن از فیلد **text** استفاده‌شده است.

```
map.addLineLayer("line", "linestring", {
  layout: { "popup-text": "The popup text." }
});
```

```
map.addLineLayer("line", "linestring", { layout: { "popup-text": "{text}" } });
```

- popup-type

این خصوصیت سبب می‌گردد تا نوع ظاهر شدن کادر اطلاعاتی را تعیین کند. مقدار پیش‌فرض آن برابر **click** است و دو مقدار **click** و **hover** را می‌تواند در بر گیرد. در نوع **click** با کلیک بروی عارضه **popup** ظاهر می‌شود و با تعیین نوع **hover** کادر اطلاعاتی به محض قرارگیری ماوس بروی عارضه، ظاهر می‌گردد. در نمونه زیر از نوع **hover** استفاده‌شده است و برای به‌کارگیری آن لازم است که از خصوصیت **popup-text** هم استفاده‌شده باشد.

```
map.addLineLayer("line", "linestring", {
  layout: { "popup-text": "{text}", "popup-type": "hover" }
});
```

- popup-offset

زمانی که از خصوصیت **icon-anchor** برای تعیین موقعیت آیکون بروی مختصات جغرافیایی مدنظر استفاده می‌شود، لازم است که جایگاه دقیق آن را برای ظاهر شدن **popup** نیز تعیین نمود. علاوه بر این خصوصیت **popup-offset** تعیین‌کننده موقعیت ظاهر شدن کادر اطلاعاتی بروی آیکون است. و این خصوصیت بر اساس تناسب آن با اندازه آیکون باید با مقادیر مناسب پر گردد مقدار باید به‌صورت آرایه‌ای از دو عدد باشد که نشان‌گر مقادیر مربوط به **x** و **y** فاصله از نقطه مدنظر است. در نمونه زیر **popup** در نقاط **20** و **30** پیکسل فاصله از مرکز یا مختصات جغرافیائی بروی آیکون نمایش می‌یابد.

```
map.addLineLayer("line", "linestring", {
  layout: { "popup-text": "{text}", "popup-offset": [20, 30] }
});
```

خصوصیت‌های paint

- line-width

پهنای خطوط را بر حسب پیکسل تعیین می‌کند و مقدار پیش‌فرض آن برابر **1** یا معادل یک پیکسل پهنا است. در نمونه کدهای زیر دو لایه با پهنای تعیین‌شده بروی نقشه رسم می‌گردد. در نمونه اول مقدار **4** یا معادل **4** پیکسل به‌صورت مستقیم برای عارضه در نظر گرفته‌شده است ولی در نمونه دیگر پهنای خط بر اساس داده پویایی که درون **properties** منبع قرار دارد تعیین‌شده است.

```
map.addLineLayer("line", "linestring", { paint: { "line-width": 4 } });
map.addLineLayer("line", "linestring", {
```

```
  paint: { "line-width": ["get", "width"] }
});
```

- line-color

رنگ خطوط را بر حسب نام یا کد رنگ تعیین می‌کند و مقدار پیش‌فرض آن برابر کد رنگ مشکی یا #000 است. در نمونه کدهای زیر دو لایه بارنگ‌های تعیین‌شده بروی نقشه ترسیم می‌گردد. در نمونه اول کد رنگ #612 به‌عنوان رنگ خطوط در نظر گرفته‌شده ولی در نمونه دیگر رنگ خط بر اساس داده پویائی که درون properties قرار دارد تعیین‌شده است.

```
map.addLineLayer("line", "linestring", { paint: { "line-color": "#612" } });
map.addLineLayer("line", "linestring", {
  paint: { "line-color": ["get", "color"] }
});
```

- line-opacity

تعیین سطح شفافیت خطوط در بازه‌ای بین 0 تا 1، به‌طوری‌که از اعداد اعشار هم می‌توان در این خصوصیت استفاده نموده و مقدار پیش‌فرض آن برابر 1 است. در نمونه کد زیر میزان سطح شفافیت خطوط برابر 0.5 است.

```
map.addLineLayer("line", "linestring ", { paint: { "line-opacity": 0.5 } });
```

خصوصیت filter

توسط فیلتر می‌توان نمایش عارضه‌ها بروی نقشه را بر اساس قاعده و شروطی خاص تعیین نمود. در شروط می‌توان از عبارات منطقی مانند >، <، =< و => نیز بهره برد. برای نمونه در کد زیر زمانی که مقدار فیلد value درون properties مربوط به منبع داده جغرافیایی کوچک‌تر و مساوی با مقدار 22 باشد فیلتر تنها بروی لایه جغرافیائی مدنظر قابل اعمال است.

```
map.addLineLayer("line", "linestring", {filter: [">=", ["get", "value"], 22]});
```

خصوصیت‌های zoom

هر لایه را می‌توان در محدوده‌ای از سطوح بزرگ‌نمائی نمایش داد که برای این کار از خصوصیت‌های minzoom و maxzoom استفاده می‌شود. در نمونه زیر لایه مدنظر تنها در بین دو سطح بزرگ‌نمائی 10 تا 18 قابل روئیت بروی نقشه است. همچنین می‌توان خصوصیت‌های مربوط به سطوح بزرگ‌نمائی را به‌صورت تکی مورداستفاده قرار داد برای نمونه با تعیین minzoom حداقل سطح بزرگ‌نمائی و با تعیین maxzoom حداکثر سطح بزرگ‌نمائی رو برای روئیت عارضه بروی نقشه تعیین نمود.

```
map.addLineLayer("line", "linestring", { minzoom: 10, maxzoom: 18 });
```

لایه fill

لایه **fill** قادر است جزئیات گرافیکی مربوط به اشکال چندضلعی را بروی نقشه نمایش دهد، داده‌هایی که از نوع **Polyon** به منبع اختصاص‌یافته‌اند برای این نوع لایه مورداستفاده قرار می‌گیرد. **fill** برای نمایش یک یا مجموعه‌ای از اشکال چندضلعی کاربرد دارند. مجموعه اشکال چندضلعی می‌بایست از نوع **MultiPolygon** درون منبع تعیین گردد.

افزودن لایه fill

توسط متد map.addFillLayer یک‌لایه از نوع **fill** به نقشه افزوده می‌شود، هر لایه **fill** حامل یک شناسه لایه و یک شناسه منبع است ازاین‌رو می‌بایست ابتدا مطابق بخش افزودن منبع منبعی خاص به نقشه افزوده شود و توسط اختصاص شناسه منبع به لایه از داده‌های آن به‌صورت نمایش بروی نقشه بهره جست.

```
map.addSource("polygon", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { value: 30 },
      geometry: {
        type: "Polygon",
        coordinates: [
          [[51.41, 35.7575], [51.4, 35.7575], [51.4, 35.75], [51.41, 35.7575]]
        ]
      }
    }
  ]
});
map.addFillLayer("fill", "polygon");
```

در نمونه کد ابتدا منبعی به نقشه با شناسه **polygon** افزوده می‌شود و حاوی داده‌ای از نوع **Polygon** است. با استفاده از متد map.addFillLayer لایه‌ای از نوع **fill** با شناسه **fill** و همچنین با استفاده از **polygon** به نقشه افزوده می‌شود. به‌طور پیش‌فرض خصوصیت مربوط به **paint** شامل **fill-color** برابر رنگ مشکی است.

استفاده از داده‌های پویا

در بخش افزودن منبع شرح داده شد که می‌توان از خصوصیت‌های **properties** درون لایه‌ها استفاده نمود. داده‌های تغییرپذیر که درون **properties** قرار می‌یابند می‌توانند دربرگیرنده مقادیری جهت کنترل **dynamic** بروی شکل چندضلعی ترسیم‌شده باشد.

```
map.addSource("polygon", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { color: "#874" },
      geometry: {
        type: "Polygon",
        coordinates: [
          [[51.41, 35.7575], [51.4, 35.7575], [51.4, 35.75], [51.41, 35.7575]]
        ]
      }
    }
  ]
});
map.addFillLayer("fill", "polygon", {
  paint: { "fill-color": ["get", "color"] }
});
```

در نمونه بالا یک‌لایه با داده‌های پویا به نقشه افزوده می‌شود داده‌های بکار گرفته‌شده در **properties** با توجه به عارضه چندضلعی مشخص‌شده حاوی یک فیلد اطلاعاتی **color** است که این فیلد را می‌توان به‌عنوان مقادیری برای خصوصیت‌های موجود مورداستفاده قرار داد. با توجه به مقدار، رنگ شکل چندضلعی ترسیم‌شده برابر مقدار داده‌شده خواهد شد که این نوع استفاده بدین معنی است که از داده‌های پویایی که در بخش **properties** مربوط به منبع داده جغرافیائی اختصاص‌یافته، استفاده نموده و فیلدهای اطلاعاتی می‌توانند با اسامی دلخواه هم مورداستفاده قرار گیرند.

گزینه‌های لایه fill

هر لایه **fill** می‌تواند شامل گزینه‌های مختلفی ازجمله **layout**، **paint**، **filter**، **minzoom** و **maxzoom** باشد. هدف استفاده از گزینه‌های لایه این است که با استفاده از آن‌ها اشکال چندضلعی خاص بروی نقشه با توجه به تعیین رنگ پس‌زمینه، رنگ حاشیه، سطح شفافیت، بعد از افزودن لایه یا بعد از تعیین **visibility** نمایش یابد. همچنین در **layout** می‌توان **popup** را تعیین نمود و در **paint** نیز می‌توان

رنگ و پهنای حاشیه شکل چندضلعی مدنظر را تعیین نموده و درنهایت در گزینه **filter** می‌توان از شروط خاصی برای نمایش لایه استفاده نمود. از گزینه‌های **minzoom** و **maxzoom** به ترتیب برای تعیین محدوده نمایش عارضه بروی نقشه استفاده می‌گردد به‌طوری‌که بر اساس تعیین محدوده سطح بزرگ‌نمائی در **zoom level** مدنظر عارضه به نمایش در خواهد آمد.

خصوصیت‌های layout

- **visibility**

این خصوصیت سبب نمایش لایه بروی نقشه می‌شود، گزینه‌های موجود در این خصوصیت شامل **visible** و **none** است و مقدار پیش‌فرض آن برابر **visible** است، به‌طوری‌که هنگام افزودن یک‌لایه بلافاصله عارضه نمایان می‌گردد می‌توان با استفاده از مقدار **none** لایه را افزوده ولی نمایش ند سپس توسط متد map.setLayerLayout خصوصیت **visiblity** را به مقدار دلخواه تغییر داد. در نمونه زیر لایه ایجاد ولی پس از افزوده شدن نمایش نمی‌یابد.

```
map.addFillLayer("fill", "polygon", { layout: { visibility: "none" } });
```

- **popup-text**

با افزودن این خصوصیت یک کادر که معمولاً حاوی اطلاعاتی در مورد عارضه مدنظر است بروی آن ظاهر می‌شود. که با کلیک بروی عارضه‌این کادر اطلاعاتی بروی آن ظاهر می‌شود. در نمونه زیر متن مدنظر با کلیک بروی عارضه بروی آن ظاهر می‌شود. این خصوصیت می‌تواند همچنین از داده‌های پویای تعیین‌شده در **properties** منبع نیز استفاده کند. در این صورت شکل ورودی با توجه به نمونه دوم تغییر می‌یابد. که در آن از فیلد **text** استفاده‌شده است.

```
map.addFillLayer("fill", "polygon", {
  layout: { "popup-text": "The popup text." }
});
map.addFillLayer("fill", "polygon", { layout: { "popup-text": "{text}" } });
```

- **popup-type**

این خصوصیت سبب می‌گردد تا نوع ظاهر شدن کادر اطلاعاتی را تعیین کند. مقدار پیش‌فرض آن برابر **click** است و دو مقدار **click** و **hover** را می‌تواند در بر گیرد. در نوع **click** با کلیک بروی عارضه **popup** ظاهر می‌شود و با تعیین نوع **hover** کادر اطلاعاتی به محض قرارگیری ماوس بروی عارضه، ظاهر می‌گردد. در نمونه زیر از نوع **hover** استفاده‌شده است و برای به‌کارگیری آن لازم است که از خصوصیت **popup-text** هم استفاده‌شده باشد.

```
map.addFillLayer("fill", "polygon", {
  layout: { "popup-text": "{text}", "popup-type": "hover" }
```

```
});
```

- popup-offset

زمانی که از خصوصیت icon-anchor برای تعیین موقعیت آیکون بروی مختصات جغرافیایی مدنظر استفاده می‌شود، لازم است که جایگاه دقیق آن را برای ظاهر شدن popup نیز تعیین نمود. علاوه بر این خصوصیت popup-offset تعیین‌کننده موقعیت ظاهر شدن کادر اطلاعاتی بروی آیکون است. و این خصوصیت بر اساس تناسب آن با اندازه آیکون باید با مقادیر مناسب پر گردد مقدار باید به‌صورت آرایه‌ای از دو عدد باشد که نشان‌گر مقادیر مربوط به x و y فاصله از نقطه مدنظر است. در نمونه زیر popup در نقاط 20 و 30 پیکسل فاصله از مرکز یا مختصات جغرافیائی بروی آیکون نمایش می‌یابد.

```
map.addFillLayer("fill", "polygon", {
  layout: { "popup-text": "{text}", "popup-offset": [20, 30] }
});
```

خصوصیت‌های paint

- fill-color

رنگ پس‌زمینه شکل چندضلعی را بر حسب نام یا کد رنگ تعیین می‌کند و مقدار پیش‌فرض آن برابر کد رنگ مشکی یا #000 است. در نمونه کدهای زیر دو لایه بارنگ‌های تعیین‌شده بروی نقشه ترسیم می‌گردد. در نمونه اول کد رنگ #612 به‌عنوان رنگ پس‌زمینه در نظر گرفته‌شده ولی در نمونه دیگر رنگ پس‌زمینه بر اساس داده پویائی که درون properties قرار دارد تعیین‌شده است.

```
map.addFillLayer("fill", "polygon", { paint: { "fill-color": "#612" } });
map.addFillLayer("fill", "polygon", {
  paint: { "fill-color": ["get", "color"] }
});
```

- fill-opacity

تعیین سطح شفافیت پس‌زمینه شکل چندضلعی در بازه‌ای بین 0 تا 1، به‌طوری‌که از اعداد اعشار هم می‌توان در این خصوصیت استفاده نموده و مقدار پیش‌فرض آن برابر 1 است. در نمونه کد زیر میزان سطح شفافیت پس‌زمینه شکل چندضلعی برابر 0.5 است.

```
map.addSymbolLayer("fill", "polygon ", { paint: { "line-opacity": 0.5 } });
```

خصوصیت filter

توسط فیلتر می‌توان نمایش عارضه‌ها بروی نقشه را بر اساس قاعده و شروطی خاص تعیین نمود. در شروط می‌توان از عبارات منطقی مانند >، <، => و =< نیز بهره برد. برای نمونه در کد زیر زمانی که مقدار فیلد **value** درون **properties** مربوط به منبع داده جغرافیایی کوچک‌تر و مساوی با مقدار **22** باشد فیلتر تنها بروی لایه جغرافیائی مدنظر قابل اعمال است.

```
map.addLineLayer("fill", "polygon", { filter: [">=", ["get", "value"], 22] });
```

خصوصیت‌های zoom

هر لایه را می‌توان در محدوده‌ای از سطوح بزرگ‌نمائی نمایش داد که برای این کار از خصوصیت‌های **minzoom** و **maxzoom** استفاده می‌شود. در نمونه زیر لایه مدنظر تنها در بین دو سطح بزرگ‌نمائی **10** تا **18** قابل روئیت بروی نقشه است. همچنین می‌توان خصوصیت‌های مربوط به سطوح بزرگ‌نمائی را به‌صورت تکی مورداستفاده قرار داد برای نمونه با تعیین **minzoom** حداقل سطح بزرگ‌نمائی و با تعیین **maxzoom** حداکثر سطح بزرگ‌نمائی رو برای روئیت عارضه بروی نقشه تعیین نمود.

```
map.addFillLayer("fill", "polygon", { minzoom: 10, maxzoom: 18 });
```

لایه circle

لایه **circle** قادر است جزئیات گرافیکی مربوط به شکل‌های دایره‌ای را بروی نقشه نمایش دهد، داده‌هایی که از نوع **Point** به منبع اختصاص‌یافته‌اند برای این نوع لایه مورداستفاده قرار می‌گیرد. **circle** برای نمایش یک یا مجموعه‌ای از دوایر کاربرد دارند.

افزودن لایه circle

توسط متد map.addCircleLayer یک‌لایه از نوع **circle** به نقشه افزوده می‌شود، هر لایه **circle** حامل یک شناسه لایه و یک شناسه منبع است ازاین‌رو می‌بایست ابتدا مطابق بخش افزودن منبع منبعی خاص به نقشه افزوده شود و توسط اختصاص شناسه منبع به لایه از داده‌های آن به‌صورت نمایش بروی نقشه بهره جست.

```
map.addSource("point", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: {},
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7575]
      }
    }
  ]
});
map.addCircleLayer("circle", "point");
```

در نمونه کد ابتدا منبعی به نقشه با شناسه **point** افزوده می‌شود و حاوی داده‌ای از نوع **Point** است. با استفاده از متد map.addCircleLayer لایه‌ای از نوع **circle** با شناسه **circle** و همچنین با استفاده از **point** به نقشه افزوده می‌شود. به‌طور پیش‌فرض خصوصیت‌های مربوط به **paint** شامل **circle-radius** برابر **5** پیکسل و **circle-color** برابر رنگ مشکی است.

استفاده از داده‌های پویا

در بخش افزودن منبع شرح داده شد که می‌توان از خصوصیت‌های **properties** درون لایه‌ها استفاده نمود. داده‌های تغییرپذیر که درون **properties** قرار می‌یابند می‌توانند دربرگیرنده مقادیری جهت کنترل **dynamic** بروی شکل دایره‌ای ترسیم‌شده باشد.

```
map.addSource("point", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { color: "#874" },
      geometry: {
        type: "Point",
        coordinates: [51.41, 35.7575]
      }
    }
  ]
});
map.addCircleLayer("circle", "point", {
  paint: { "circle-color": ["get", "color"] }
});
```

در نمونه بالا یک‌لایه با داده‌های پویا به نقشه افزوده می‌شود داده‌های بکار گرفته‌شده در **properties** با توجه به عارضه دایره‌ای شکل مشخص‌شده حاوی یک فیلد اطلاعاتی **color** است که این فیلد را می‌توان به‌عنوان مقادیری برای خصوصیت‌های موجود مورداستفاده قرار داد. با توجه به مقدار، رنگ پس‌زمینه شکل دایره‌ای ترسیم‌شده برابر مقدار داده‌شده خواهد شد که این نوع استفاده بدین معنی است که از داده‌های پویایی که در بخش **properties** مربوط به منبع داده جغرافیائی اختصاص‌یافته، استفاده نموده و فیلدهای اطلاعاتی می‌توانند با اسامی دلخواه هم مورداستفاده قرار گیرند.

گزینه‌های لایه circle

هر لایه **circle** می‌تواند شامل گزینه‌های مختلفی ازجمله **layout**، **paint**، **filter**، **minzoom** و **maxzoom** باشد. هدف استفاده از گزینه‌های لایه این است که با استفاده از آن‌ها اشکال دایره‌ای خاص بروی نقشه با توجه به تعیین رنگ پس‌زمینه، سطح شفافیت، بعد از افزودن لایه یا بعد از تعیین **visibility** نمایش یابد. همچنین در **layout** می‌توان **popup** را تعیین نمود و در **paint** نیز می‌توان رنگ و پهنای حاشیه شکل چندضلعی مدنظر را تعیین نموده و درنهایت در گزینه **filter** می‌توان از شروط خاصی برای

نمایش لایه استفاده نمود. از گزینه‌های **minzoom** و **maxzoom** به ترتیب برای تعیین محدوده نمایش عارضه بروی نقشه استفاده می‌گردد به‌طوری‌که بر اساس تعیین محدوده سطح بزرگ‌نمائی در **zoom level** مدنظر عارضه به نمایش در خواهد آمد.

خصوصیت‌های layout

- **visibility**

این خصوصیت سبب نمایش لایه بروی نقشه می‌شود، گزینه‌های موجود در این خصوصیت شامل **visible** و **none** است و مقدار پیش‌فرض آن برابر **visible** است، به‌طوری‌که هنگام افزودن یک‌لایه بلافاصله عارضه نمایان می‌گردد می‌توان با استفاده از مقدار **none** لایه را افزوده ولی نمایش ند سپس توسط متد map.setLayerLayout خصوصیت **visiblity** را به مقدار دلخواه تغییر داد. در نمونه زیر لایه ایجاد ولی پس از افزوده شدن نمایش نمی‌یابد.

```
map.addCircleLayer("circle", "point", { layout: { visibility: "none" } });
```

- **popup-text**

با افزودن این خصوصیت یک کادر که معمولاً حاوی اطلاعاتی در مورد عارضه مدنظر است بروی آن ظاهر می‌شود. که با کلیک بروی عارضه‌این کادر اطلاعاتی بروی آن ظاهر می‌شود. در نمونه زیر متن مدنظر با کلیک بروی عارضه بروی آن ظاهر می‌شود. این خصوصیت می‌تواند همچنین از داده‌های پویای تعیین‌شده در **properties** منبع نیز استفاده کند. در این صورت شکل ورودی با توجه به نمونه دوم تغییر می‌یابد. که در آن از فیلد **text** استفاده‌شده است.

```
map.addCircleLayer("circle", "point", {
  layout: { "popup-text": "The popup text." }
});
map.addCircleLayer("circle", "point", { layout: { "popup-text": "{text}" } });
```

- **popup-type**

این خصوصیت سبب می‌گردد تا نوع ظاهر شدن کادر اطلاعاتی را تعیین کند. مقدار پیش‌فرض آن برابر **click** است و دو مقدار **click** و **hover** را می‌تواند در بر گیرد. در نوع **click** با کلیک بروی عارضه **popup** ظاهر می‌شود و با تعیین نوع **hover** کادر اطلاعاتی به محض قرارگیری ماوس بروی عارضه، ظاهر می‌گردد. در نمونه زیر از نوع **hover** استفاده‌شده است و برای به‌کارگیری آن لازم است که از خصوصیت **popup-text** هم استفاده‌شده باشد.

```
map.addCircleLayer("circle", "point", {
  layout: { "popup-text": "{text}", "popup-type": "hover" }
```

```
});
```

- **popup-offset**

زمانی که از خصوصیت **icon-anchor** برای تعیین موقعیت آیکون بروی مختصات جغرافیایی مدنظر استفاده می‌شود، لازم است که جایگاه دقیق آن را برای ظاهر شدن **popup** نیز تعیین نمود. علاوه بر این خصوصیت **popup-offset** تعیین‌کننده موقعیت ظاهر شدن کادر اطلاعاتی بروی آیکون است. و این خصوصیت بر اساس تناسب آن با اندازه آیکون باید با مقادیر مناسب پر گردد مقدار باید به‌صورت آرایه‌ای از دو عدد باشد که نشان‌گر مقادیر مربوط به **x** و **y** فاصله از نقطه مدنظر است. در نمونه زیر **popup** در نقاط **20** و **30** پیکسل فاصله از مرکز یا مختصات جغرافیائی بروی آیکون نمایش می‌یابد.

```
map.addCircleLayer("circle", "point", {
  layout: { "popup-text": "{text}", "popup-offset": [20, 30] }
});
```

خصوصیت‌های paint

- **circle-color**

رنگ پس‌زمینه شکل دایره‌ای را بر حسب نام یا کد رنگ تعیین می‌کند و مقدار پیش‌فرض آن برابر کد رنگ مشکی یا **#000** است. در نمونه کدهای زیر دو لایه بارنگ‌های تعیین‌شده بروی نقشه ترسیم می‌گردد. در نمونه اول کد رنگ **#612** به‌عنوان رنگ پس‌زمینه در نظر گرفته‌شده ولی در نمونه دیگر رنگ پس‌زمینه بر اساس داده پویائی که درون **properties** قرار دارد تعیین‌شده است.

```
map.addCircleLayer("circle", "point", { paint: { "circle-color": "#612" } });
map.addCircleLayer("circle", "point", {
  paint: { "circle-color": ["get", "color"] }
});
```

- **circle-opacity**

تعیین سطح شفافیت پس‌زمینه شکل دایره‌ای در بازه‌ای بین **0** تا **1**، به‌طوری‌که از اعداد اعشار هم می‌توان در این خصوصیت استفاده نموده و مقدار پیش‌فرض آن برابر **1** است. در نمونه کد زیر میزان سطح شفافیت پس‌زمینه شکل دایره‌ای برابر **0.5** است.

```
map.addCircleLayer("circle", "point ", { paint: { "circle-opacity": 0.5 } });
```

- **circle-radius**

به‌وسیله تعیین این خصوصیت می‌توان شعاع دایره را مشخص کرد. شعاع دایره بر حسب پیکسل تعیین می‌گردد و مقدار پیش‌فرض آن برابر **5** پیکسل است. شعاع دایره در سطوح مختلف بزرگ‌نمائی ثابت می‌ماند و اغلب برای نمایش جزئیات گرافیکی ثابت به کار گرفته می‌شود. در

نمونه‌های زیر سه لایه ایجاد می‌گردند. اولین لایه بر اساس مقدار ورودی دایره‌ای با شعاع **10** پیکسل ترسیم می‌کند. در لایه بعدی شعاع دایره از داده‌های پویا **properties** گرفته می‌شود. شعاع دایره می‌تواند در رنج‌های مختلف متفاوت باشد در لایه سوم بر اساس بازه‌ای که تعریف شده شعاع دایره نیز بر اساس آن ترسیم می‌شود به‌طوری‌که از سطح بزرگ‌نمائی **1** تا **14** برابر **5** پیکسل و از سطح بزرگ‌نمائی **18** به بعد شعاع دایره ترسیم‌شده برابر **15** پیکسل خواهد بود.

```
map.addCircleLayer("circle", "point", { paint: { "circle-radius": 10 } });
map.addCircleLayer("circle", "point", {
  paint: { "circle-radius": ["get", "radius"] }
});
map.addCircleLayer("circle", "point", {
  paint: {
    "circle-radius": ["interpolate", ["linear"], ["zoom"],
        14, 5,
        18, 15
    ]
  }
});
```

- `circle-scalable-raidus`

به‌وسیله تعیین این خصوصیت می‌توان شعاع دایره را به‌صورت مقیاس‌پذیر مشخص کرد. شعاع دایره بر حسب متر تعیین می‌گردد و با تغییر در سطح بزرگ‌نمائی دایره نیز بر حسب متر در پیکسل تغییر می‌کند. ازاین‌رو برای نمایش حقیقی یا واقعی یک محدوده روی نقشه در شعاع دایره، می‌توان از آن بهره برد. در نمونه زیر دو لایه به صورتی در نقشه نمایش می‌یابند که لایه اول دایره‌ای با شعاع **20** متر بوده و در لایه دوم شعاع بر اساس داده‌های پویایی که در **properties** منبع قرار دارد، تعیین می‌گردد.

```
map.addCircleLayer("circle", "point", {
  paint: { "circle-scalable-radius": 20 }
});
map.addCircleLayer("circle", "point", {
  paint: { "circle-scalable-radius": ["get", "radius"] }
});
```

خصوصیت filter

توسط فیلتر می‌توان نمایش عارضه‌ها بروی نقشه را بر اساس قاعده و شروطی خاص تعیین نمود. در شروط می‌توان از عبارات منطقی مانند >، <، => و =< نیز بهره برد. برای نمونه در کد زیر زمانی که مقدار فیلد **value** درون **properties** مربوط به منبع داده جغرافیایی کوچک‌تر و مساوی با مقدار **22** باشد فیلتر تنها بروی لایه جغرافیائی مدنظر قابل اعمال است.

```
map.addCircleLayer("circle", "point", { filter: [">=", ["get", "value"], 22] });
```

خصوصیت‌های zoom

هر لایه را می‌توان در محدوده‌ای از سطوح بزرگ‌نمائی نمایش داد که برای این کار از خصوصیت‌های **minzoom** و **maxzoom** استفاده می‌شود. در نمونه زیر لایه مدنظر تنها در بین دو سطح بزرگ‌نمائی **10** تا **18** قابل روئیت بروی نقشه است. همچنین می‌توان خصوصیت‌های مربوط به سطوح بزرگ‌نمائی را به‌صورت تکی مورداستفاده قرار داد برای نمونه با تعیین **minzoom** حداقل سطح بزرگ‌نمائی و با تعیین **maxzoom** حداکثر سطح بزرگ‌نمائی رو برای روئیت عارضه بروی نقشه تعیین نمود.

```
map.addCircleLayer("circle", "point", { minzoom: 10, maxzoom: 18 });
```

لایه heatmap

لایه **heatmap** قادر است که جزئیات گرافیکی مربوط به مجموعه‌ای از نقاط را بر روی نقشه توسط پوششی خاص نمایش دهد، منظور از پوشش ، نواحی رنگی است که هنگامی‌که نقاط در تراکم زیادی قرار دارند به‌طوری‌که سطح بزرگ‌نمائی در پائین ترین حد ممکن خود باشد پیوسته بودن نقاط در محدوده‌ها توسط پوشش رنگی مایل به رنگ قرمز نمایش می‌یابد.

افزودن لایه heatmap

توسط متد map.addHeatmapLayer یک‌لایه از نوع **heatmap** به نقشه افزوده می‌شود، هر لایه **heatmap** حامل یک شناسه لایه و یک شناسه منبع است ازاین‌رو می‌بایست ابتدا مطابق بخش افزودن منبع منبعی خاص به نقشه افزوده شود و توسط اختصاص شناسه منبع به لایه از داده‌های آن به‌صورت نمایش بروی نقشه بهره جست.

```
map.addSource("points", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      properties: { value: 2 },
      geometry: { type: "Point", coordinates: [51.27, 35.09] }
    },
    {
      type: "Feature",
      properties: { value: 0 },
      geometry: { type: "Point", coordinates: [51.09, 35.54] }
    },
    {
      type: "Feature",
      properties: { value: 5 },
      geometry: { type: "Point", coordinates: [51.27, 35.54] }
    },
    {
      type: "Feature",
      properties: { value: 2 },
      geometry: { type: "Point", coordinates: [51.36, 35.542] }
    },
    {
```

```
      type: "Feature",
      properties: { value: 1 },
      geometry: { type: "Point", coordinates: [51.45, 35.63] }
    },
    {
      type: "Feature",
      properties: { value: 4 },
      geometry: { type: "Point", coordinates: [51, 35.81] }
    }
  ]
});
map.addHeatmapLayer('heatmap', 'points', {
    maxzoom: 9,
    paint: {
        'heatmap-weight': ['interpolate', ['linear'], ['get', 'value'],
            0, 0,
            6, 1,
        ],
        'heatmap-intensity': ['interpolate', ['linear'], ['zoom'],
            0, 1,
            9, 3,
        ],
        'heatmap-color': ['interpolate', ['linear'], ['heatmap-density'],
            0, 'rgba(33,102,172,0)',
            0.2, 'rgb(103,169,207)',
            0.4, 'rgb(209,229,240)',
            0.6, 'rgb(253,219,199)',
            0.8, 'rgb(239,138,98)',
            1, 'rgb(178,24,43)',
        ],
        'heatmap-radius': ['interpolate', ['linear'], ['zoom'],
            0, 2,
            9, 20,
        ],
        'heatmap-opacity': ['interpolate', ['linear'], ['zoom'],
            7, 1,
            9, 0,
        ],
    },
});
```

در نمونه کد بالا پوشش رنگی در نقاطی که مقدار **value** داده پویای آن‌ها نزدیک به **6** باشد، مایل به رنگ قرمز در خواهد آمد و رنگ نواحی از طریق تقسیم‌بندی به **6** بازه در اصل برای نمایش تراکم و رنگی پوشش نواحی خاص است.

گزینه‌های لایه heatmap

هر لایه **heatmap** می‌تواند شامل گزینه‌های مختلفی ازجمله **layout**، **paint**، **filter**، **minzoom** و **maxzoom** باشد. هدف استفاده از گزینه‌های لایه این است که با استفاده از آن‌ها پوشش رنگی خاص بروی نقشه با توجه به تعیین پهنا، تراکم، رنگ، شعاع و سطح شفافیت بعد از افزودن لایه یا بعد از تعیین **visibility** نمایش یابد. همچنین در **layout** می‌توان **popup** را تعیین نمود و در **paint** نیز می‌توان رنگ و سایر خصوصیت‌های گرافیکی را تعیین نموده و درنهایت در گزینه **filter** می‌توان از شروط خاصی برای نمایش لایه استفاده نمود. از گزینه‌های **minzoom** و **maxzoom** به ترتیب برای تعیین محدوده نمایش عارضه بروی نقشه استفاده می‌گردد به‌طوری‌که بر اساس تعیین محدوده سطح بزرگ‌نمائی در **zoom level** مدنظر عارضه به نمایش در خواهد آمد.

خصوصیت‌های layout

- **visibility**

این خصوصیت سبب نمایش لایه بروی نقشه می‌شود، گزینه‌های موجود در این خصوصیت شامل **visible** و **none** است و مقدار پیش‌فرض آن برابر **visible** است، به‌طوری‌که هنگام افزودن یک‌لایه بلافاصله عارضه نمایان می‌گردد می‌توان با استفاده از مقدار **none** لایه را افزوده ولی نمایش ند سپس توسط متد map.setLayerLayout خصوصیت **visiblity** را به مقدار دلخواه تغییر داد. در نمونه زیر لایه ایجادشده ولی پس از افزوده شدن نمایش نمی‌یابد.

```
map.addHeatmapLayer("heatmap", "points", {
  paint: {
    "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
      0, "rgba(33,102,172,0)",
      1, "rgb(178,24,43)"
    ]
  },
  layout: { visibility: "visible" }
});
```

- **popup-text**

با افزودن این خصوصیت یک کادر که معمولاً حاوی اطلاعاتی در مورد عارضه مدنظر است بروی آن ظاهر می‌شود. که با کلیک بروی عارضه‌این کادر اطلاعاتی بروی آن ظاهر می‌شود. در نمونه زیر متن مدنظر با کلیک بروی عارضه بروی آن ظاهر می‌شود. این خصوصیت می‌تواند همچنین از داده‌های پویای تعیین‌شده در **properties** منبع نیز استفاده کند. در این صورت شکل ورودی با توجه به نمونه دوم تغییر می‌یابد. که در آن از فیلد **text** استفاده‌شده است.

```
map.addHeatmapLayer("heatmap", "points", {
    paint: {
      "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
        0, "rgba(33,102,172,0)",
        1, "rgb(178,24,43)"
      ]
    },
    layout: { "popup-text": "The popup text." }
});
map.addHeatmapLayer("heatmap", "points", {
    paint: {
      "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
        0, "rgba(33,102,172,0)",
        1, "rgb(178,24,43)"
      ]
    },
    layout: { "popup-text": "{text}" }
});
```

- popup-type

این خصوصیت سبب می‌گردد تا نوع ظاهر شدن کادر اطلاعاتی را تعیین کند. مقدار پیش‌فرض آن برابر **click** است و دو مقدار **click** و **hover** را می‌تواند در بر گیرد. در نوع **click** با کلیک بروی عارضه **popup** ظاهر می‌شود و با تعیین نوع **hover** کادر اطلاعاتی به محض قرارگیری ماوس بروی عارضه، ظاهر می‌گردد. در نمونه زیر از نوع **hover** استفاده‌شده است و برای به‌کارگیری آن لازم است که از خصوصیت **popup-text** هم استفاده‌شده باشد.

```
map.addHeatmapLayer("heatmap", "points", {
    paint: {
      "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
        0, "rgba(33,102,172,0)",
        1, "rgb(178,24,43)"
      ]
```

```
    },
    layout: { "popup-text": "{text}", "popup-type": "hover" }
});
```

- popup-offset

زمانی که از خصوصیت icon-anchor برای تعیین موقعیت آیکون بروی مختصات جغرافیایی مدنظر استفاده می‌شود، لازم است که جایگاه دقیق آن را برای ظاهر شدن popup نیز تعیین نمود. علاوه بر این خصوصیت popup-offset تعیین‌کننده موقعیت ظاهر شدن کادر اطلاعاتی بروی آیکون است. و این خصوصیت بر اساس تناسب آن با اندازه آیکون باید با مقادیر مناسب پر گردد مقدار باید به‌صورت آرایه‌ای از دو عدد باشد که نشان‌گر مقادیر مربوط به x و y فاصله از نقطه مدنظر است. در نمونه زیر popup در نقاط 20 و 30 پیکسل فاصله از مرکز یا مختصات جغرافیائی بروی آیکون نمایش می‌یابد.

```
map.addHeatmapLayer("heatmap", "points", {
    paint: {
      "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
        0, "rgba(33,102,172,0)",
        1, "rgb(178,24,43)"
      ]
    },
    layout: { "popup-text": "{text}", "popup-offset": [20, 30] }
});
```

خصوصیت‌های paint

- heatmap-weight

پهنای نواحی پوششی را بر حسب پیکسل تعیین می‌کند. در نمونه کد بازه‌ای خاص از اعداد 0 تا 6 تعیین‌کننده پهنای نواحی است به‌طوری‌که با استفاده از داده پویا در فیلد اطلاعاتی value از properties منبع هرگاه عدد 0 تا 6 بود پهنای نواحی برابر 0 و هرگاه عدد 6 به بالا بود پهنای نواحی برابر عدد 1 خواهد بود.

```
map.addHeatmapLayer('heatmap', 'points', {
    paint: {
        'heatmap-weight': ['interpolate', ['linear'], ['get', 'value'],
            0, 0,
            6, 1,
        ]
    }
```

```
});
```

- heatmap-intensity

شدت پوشش نواحی توسط این خصوصیت تعیین می‌گردد برای نمونه در کد زیر شد نواحی در سطوح مختلف بزرگ‌نمائی برای مثال 0 برابر 1 و در سطح بزرگ‌نمائی 9 برابر 3 واحد است.

```
map.addHeatmapLayer('heatmap', 'points', {
    paint: {
        'heatmap-intensity': ['interpolate', ['linear'], ['zoom'],
            0, 1,
            9, 3,
        ],
    },
});
```

- heatmap-color

رنگی نواحی را می‌تواند بر اساس تراکم در نواحی خاصی از نقشه تعیین کند. در این حالت میزان سطح تراکم بین عدد 0 تا 1 در نظر گرفته می‌شود که به این معنی است که 0 برابر کمترین میزان و 1 برابر بیشترین میزان تراکم در ناحیه است، منظور از تراکم در ناحیه مشخص این است که در ناحیه مذکور چه مقدار نقاطی وجود دارند. در نمونه کد برای 6 حالت مختلف از تراکم رنگ‌های متنوعی در نظر گرفته‌شده که درنهایت در بیشترین تراکم به رنگ قرمز متمایل می‌گردد.

```
map.addHeatmapLayer('heatmap', 'points', {
    maxzoom: 9,
    paint: {
        'heatmap-color': ['interpolate', ['linear'], ['heatmap-density'],
            0, 'rgba(33,102,172,0)',
            0.2, 'rgb(103,169,207)',
            0.4, 'rgb(209,229,240)',
            0.6, 'rgb(253,219,199)',
            0.8, 'rgb(239,138,98)',
            1, 'rgb(178,24,43)',
        ],
    },
});
```

- heatmap-radius

شعاع دوایر را در نواحی خاص تعیین می‌کند و بر حسب پیکسل است. در نمونه کد زیر در حالتی که سطح بزرگ‌نمائی از 0 تا 9 است شعاع نواحی برابر 2 پیکسل و درصورتی‌که سطح بزرگ‌نمائی از 9 به بالا باشد میزان شعاع برابر 20 پیکسل خواهد بود.

```
map.addHeatmapLayer('heatmap', 'points', {
```

```
    paint: {
        'heatmap-radius': ['interpolate', ['linear'], ['zoom'],
            0, 2,
            9, 20,
        ],
    },
});
```

- heatmap-opacity

میزان سطح شفافیت را در نواحی خاص تعیین می‌کند به‌طوری‌که میزان سطح شفافیت در بازه‌ای بین **0** تا **1** است. در نمونه کد میزان سطح شفافیت در سطح بزرگ‌نمائی **7** تا **9** برابر **1** و در سطح بزرگ‌نمائی **9** به بالا برابر **0** یا غیر قابل مشاهده است. سطح شفافیت به تناسب سطح بزرگ‌نمائی پائین تر باید غیر قابل مشاهده باشند به‌طوری‌که در سطوح پایین تر میزان آشکار بودن نواحی به علت دور بودن از سطح زمین می‌بایست به عدد **1** نزدیک‌تر باشد.

```
map.addHeatmapLayer('heatmap', 'points', {
    paint: {
        'heatmap-opacity': ['interpolate', ['linear'], ['zoom'],
            7, 1,
            9, 0,
        ],
    },
});
```

خصوصیت filter

توسط فیلتر می‌توان نمایش عارضه‌ها بروی نقشه را بر اساس قاعده و شروطی خاص تعیین نمود. در شروط می‌توان از عبارات منطقی مانند >، <، =< و >= نیز بهره برد. برای نمونه در کد زیر زمانی که مقدار فیلد **value** درون **properties** مربوط به منبع داده جغرافیایی کوچک‌تر و مساوی با مقدار **22** باشد فیلتر تنها بروی لایه جغرافیائی مدنظر قابل اعمال است.

```
map.addHeatmapLayer("heatmap", "points", {
  paint: {
    "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
      0, "rgba(33,102,172,0)",
      1, "rgb(178,24,43)"
    ]
  },
  filter: [">=", ["get", "value"], 22]
```

```
});
```

خصوصیت‌های zoom

هر لایه را می‌توان در محدوده‌ای از سطوح بزرگ‌نمائی نمایش داد که برای این کار از خصوصیت‌های **minzoom** و **maxzoom** استفاده می‌شود. در نمونه زیر لایه مدنظر تنها در بین دو سطح بزرگ‌نمائی **10** تا **18** قابل روئیت بروی نقشه است. همچنین می‌توان خصوصیت‌های مربوط به سطوح بزرگ‌نمائی را به‌صورت تکی مورداستفاده قرار داد برای نمونه با تعیین **minzoom** حداقل سطح بزرگ‌نمائی و با تعیین **maxzoom** حداکثر سطح بزرگ‌نمائی رو برای روئیت عارضه بروی نقشه تعیین نمود.

```
map.addHeatmapLayer("heatmap", "points", {
  paint: {
    "heatmap-color": ["interpolate", ["linear"], ["heatmap-density"],
      0, "rgba(33,102,172,0)",
      1, "rgb(178,24,43)"
    ]
  },
  minzoom: 10,
  maxzoom: 18
});
```

خوشه‌بندی

زمانی که تراکم عارضه‌ها از نوع Point بروی نقشه زیاد باشد و یا برای نمایش مجموعه‌ای وسیع از داده‌ها بروی نقشه می‌توان از خوشه‌بندی استفاده نمود. در clustering داده‌های قابل‌نمایش در مجموعه‌ای از دوایر قرار می‌گیرند و بر اساس گروه‌بندی از همدیگر جدا و بخش زیادی از داده در خوشه‌های مرتبط قرار می‌گیرند ازاین‌رو مدیریت و نمایش داده‌ها بهتر انجام خواهد گرفت.

منبع جهت خوشه‌بندی

توسط متد map.addSource منبعی می‌تواند به نقشه افزوده شود و توسط گزینه‌های منبع یا سومین پارامتر ورودی، می‌توان خصوصیت‌های cluster را تعیین نمود.

گزینه‌های منبع جهت خوشه‌بندی

گزینه cluster

هر منبع می‌تواند توسط این گزینه بدون استفاده از خصوصیت‌ها نیز می‌توان منبع مدنظر را به‌صورت خوشه‌بندی بروی نقشه استفاده نمود.

- maxZoom

توسط این خصوصیت حداکثر سطح بزرگ‌نمائی جهت مشاهده خوشه‌بندی بروی نقشه تعیین می‌گردد. در نمونه کد حداکثر سطح بزرگ‌نمائی جهت نمایش خوشه‌بندی برابر 14 است.

```
map.addSource('points', geoJson, { cluster: { maxZoom: 14 } });
```

- radius

این خصوصیت شعاع محدوده خوشه‌بندی را بر حسب متر تعیین می‌کند و مقدار پیش‌فرض آن برابر 50 متر است. منظور از شعاع، محدوده‌ای است که داده‌های موجود در آن نواحی درون یک گروه خاص قرار می‌یابد برای نمونه تمامی نقاطی که در شعاع 50 متری هم باشند به یک خوشه تبدیل می‌شوند و نواحی دیگر نیز تشکیل خوشه‌های دیگری می‌دهند. در نمونه کد زیر خوشه‌بندی بر اساس شعاع 80 متری انجام می‌گیرد. باید توجه داشت که هرچقدر شعاع دایره خوشه‌بندی بیشتر باشد میزان خوشه‌ها کاهش می‌یابد ازاین‌رو این عدد می‌بایست با تراکم نقاط تناسب داشته باشد تا خوشه‌بندی مناسبی بروی داده‌ها انجام گیرد.

```
map.addSource('points', geoJson, { cluster: { radius: 80 } });
```

اعمال خوشه‌بندی

افزودن منبع خوشه‌بندی

جهت اعمال خوشه‌بندی ابتدا نیاز به افزودن یک منبع و اختصاص گزینه **cluster** به منبع است. در کد زیر خوشه‌بندی بروی **6** داده از نوع **Point** انجام‌گرفته و شعاع برای خوشه‌بندی برابر **25** متر و حداکثر سطح بزرگ‌نمائی هم **14** در نظر گرفته‌شده است. برای نمایش بهتر خوشه‌بندی نیاز به در دست داشتن داده‌های جغرافیائی بیشتر از تعدادی است که در مثال قرار دارد و این تنها نمونه‌ای ساده از خوشه‌بندی خواهد بود.

```
map.addSource("points", {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      geometry: { type: "Point", coordinates: [51.27, 35.09] }
    },
    {
      type: "Feature",
      geometry: { type: "Point", coordinates: [51.09, 35.54] }
    },
    {
      type: "Feature",
      geometry: { type: "Point", coordinates: [51.27, 35.54] }
    },
    {
      type: "Feature",
      geometry: { type: "Point", coordinates: [51.36, 35.542] }
    },
    {
      type: "Feature",
      geometry: { type: "Point", coordinates: [51.45, 35.63] }
    },
    {
      type: "Feature",
      geometry: { type: "Point", coordinates: [51, 35.81] }
    }
  ]
}, { cluster: { radius: 25, maxZoom: 14 } });
```

افزودن لایه خوشه‌ها

جهت نمایش داده‌های خوشه‌بندی بروی نقشه می‌توان از متد map.addCircleLayer استفاده کرد. در نمونه کد زیر خوشه‌ها بر اساس بازه‌ای از سه رنگ بروی نقشه نمایان خواهد گردید. در نمونه کد دوایر ترسیم‌شده بروی نقشه بر اساس فیلد محاسباتی point_count یا تعداد نقاطی که در خوشه قرار می‌گیرد در سه رنگ آبی با تعداد نقاط کمتر از 2، زرد با تعداد نقاط بین 2 تا 4 و صورتی با تعداد نقاط 4 و بیشتر از آن به نمایش در خواهند آمد. شعاع دوایر خوشه‌بندی به سه بازه تقسیم‌بندی شده و این سه بازه نیز با استفاده از تعداد نقاط موجود در نواحی تعیین‌شده است. با استفاده از خصوصیت cluster-expansion مربوط به layout لایه می‌توان تعیین نمود که با کلیک بروی خوشه‌ها میزان سطح بزرگ‌نمائی بیشتر شود و حالتی شبیه به تمرکز روی خوشه‌بندی دارد.

```
map.addCircleLayer("clusters", "points", {
  filter: ["has", "point_count"],
  paint: {
    "circle-color": ["step",["get", "point_count"],
      "#51bbd6",
      2, "#f1f075",
      4, "#f28cb1"
    ],
    "circle-radius": ["step", ["get", "point_count"],
    20, 2,
    30, 4,
    40
  ]},
  layout: { "cluster-expansion": "zoom" }
});
```

افزودن لایه تعداد نقاط

جهت نمایش تعداد نقاط موجود در هر خوشه بروی نقشه می‌توان از متد map.addSymbolLayer استفاده کرد. در نمونه کد زیر تعداد نقاط مربوط به هر خوشه بروی دوایر مربوط به آن نمایش می‌یابد. فیلد داده‌ای برای نمایش متن می‌بایست برابر point_count_abbreviated که این فیلد محاسباتی دربرگیرنده تعداد نقاط موجود در هر خوشه است، اندازه متن نیز در این نمونه کد برابر 12 پیکسل در نظر گرفته‌شده، شرط نمایش این لایه این است که داده حاوی تعداد پوینت یا فیلد محاسباتی point_count باشد.

```
map.addSymbolLayer("cluster-count", "points", {
```

```
  layout: {
    "text-field": "{point_count_abbreviated}",
    "text-size": 12
  },
  filter: ["has", "point_count"]
});
```

افزودن لایه نقاط خارج از خوشه‌بندی

جهت نمایش داده‌های خارج از خوشه‌بندی بروی نقشه می‌توان از متد map.addCircleLayer و یا map.addSymbolLayer با توجه به نیاز استفاده کرد. هرچقدر سطح بزرگ‌نمائی بیشتر شود خوشه‌بندی نیز محدودتر می‌شود ازاین‌رو اطلاعات نمایش یافته در سطوح بالائی از بزرگ‌نمائی همان داده‌های اصلی است که باید بر اساس نوع نیاز بروی نقشه نمایش یابند این نمایش می‌تواند یک دایره یا مارکری بروی نقشه باشد. در کد زیر پس‌ازاین که سطح بزرگ‌نمائی افزایش‌یافته و نقاط از خوشه‌بندی خارج شوند ازآن‌پس بروی نقشه پدیدار خواهند شد. خصوصیت‌های مربوط به **paint** قادر است که دایره‌ای با رنگ آبی و حاشیه‌ای سفید و شعاعی به اندازه **4** پیکسل و حاشیه‌ای به اندازه **1** پیکسل رسم کند، و شرط نمایش آن زمانی است که داده‌ها فاقد فیلد محاسباتی **point_count** باشد. بدین معنی که تعداد نقاط درون خوشه‌بندی برابر صفر باشد.

```
map.addCircleLayer("unclustered-point", "points", {
  paint: {
    "circle-color": "#11b4da",
    "circle-radius": 4,
    "circle-stroke-width": 1,
    "circle-stroke-color": "#fff"
  },
  filter: ["!", ["has", "point_count"]]
});
```