Mapbox GL JS

Mapbox GL JS는 웹페이지에서 〈canvas〉 엘리먼트와 WebGL을 이용해 인터랙티브한 벡터 지도를 렌더링하는 자바스크립트 라이브러리입니다. 크게 벡터 타일 렌더링 라이브러리와 (스크롤/줌을 지원하고 마커나 오버레이 등을 지도에 추가하게 해주는) 웹 지도 라이브러리로 구성됩니다.

자바스크립트에서 Map 객체를 생성하면 Mapbox GL JS는 WebWorker 스레드 상에서 명세된 스타일을 바탕으로 타일 데이터를 렌더링에 적합하게 재조합하고 데이터를 어떤 방식(점, 선, 라벨, 폴리곤 등)과 모양(색, 패턴, 선 두께 등)으로 렌더링할지와 그 순서를 정한 후 WebGL을 통해 렌더링합니다. 이후 Map 객체는 유저 인터페이스나 이벤트를 통해 실시간으로 변경될 수 있습니다.

웹지도 배경지식

projection: 지도투영법. 지구는 구체이기 때문에 지구의 좌표를 평면상에 옮겼을때 왜곡이 필연적으로 발생합니다. 투영법은 이 왜곡을 처리하는 방법입니다.

Mercator projection: 메르카토르 도법. 원통중심도법과 원통정적도법을 절충한 도법. 경선의 간격은 고정되어 있으나 위선의 간격을 조절하여 각도관계가 정확합니다. 이 도법을 바탕으로한 Web Mercator projection이 웹 지도의 표준으로 쓰이고 있습니다.

Zoom Level & Tile Grid: 평면 지도를 줌레벨 z에 따라 x축 y축 각 2^Z개의 바둑판 타일로 분할하여 저장하는 방식입니다.

타일(source)

Mapbox GL JS가 벡터타일을 렌더링하기 위해서는 벡터 타일 세트와 스타일 명세가 필요합니다. 레스터 타일 세트는 레스터 이미지 데이터를, 벡터 타일 세트는 GIS 소스 데이터를 줌레벨 z과 x, y 좌표 단위로 정리해놓은 바이너리 데이터 세트입니다. 레스터 타일은 위성지도나 힐쉐이드 지도 등에 사용됩니다. 벡터 타일은 실시간 인터랙션이 필요한 추상화된 그래픽 지도에 사용됩니다. Mapbox Vector Tile(mvt) 포맷은 타일 안에 layer를 정의할 수 있고, 그 레이어 안에 feature를 넣을 수 있습니다. feature는 geometries과 tag 등으로 구성됩니다.

스타일

스타일은 타일에 정의된 레이어 단위로 데이터가 렌더링될 형식과 모양을 명세한 형식입니다. Mapbox Style Specification은 source가 되는 타일의 레이어의 feature들을 불러와 tag의 값을 바탕으로 필터, 연산하여 렌더링 여부와 스타일 속성(line_width, color..)을 정할 수 있습니다. 따라서 원본 타일을 변경하지 않고도 사용자가 자유롭게 지도를 스타일링 할 수 있습니다.

타일(parsed)

우선 현재 그리고자 하는 tramsform 데이터 (위도, 경도, 줌레벨, 기울어진 각도, 돌린 각도 등)을 바탕으로 source 타일을 다운로드합니다. 준비되어있던 styleLayer에 맞춰 필요한 source 타일 레이어를 style에 명시된 방식으로 변환합니다. 이렇게 변환된 레이어는 bucket 이라는 형태로 파싱된 타일 안에 저장됩니다.

주요 객체

주요 객체들은 observer pattern을 따릅니다. 데이터 로드 및 UI 인터렉션 등의 이벤트는 Evented 객체로 관리되며, Map, Style, StyleLayer, Source는 이 Evented 를 상속받아 서로 소통합니다.

스타일 명세과 타일 다운로드와 파싱, 렌더링은 모두 브라우저의 스레드와는 병렬적으로 일어나야하기 때문에 Mapbox GL JS은 Worker api를 이용합니다. Worker 스레드를 생성하고 이와 통신하기 위한 객체로 Dispatcher와 Actor가 있습니다.

Map은 사용자가 접근할 수 있는 인터페이스 역할을 합니다. Map은 크게 Style과 Painter 를 속성으로 갖습니다. Style은 명세된 스타일과 타일 세트를 바탕으로 렌더링에 적합한 Tile 객체를 준비하는 역할을 맡고, Painter는 이를 WebGL이 렌더링할 수 있게 알맞은 Program과 Shader를 세팅하여 렌더링하는 역할입니다.

Style은 스타일 명세의 각각의 레이어들을 관장하는 StyleLayer (fill, circle, line...)와 그 StyleLayer를 렌더링할때 필요한 데이터들(Tile, Image, Glyph..)을 관장하는 SourceCache를 관리합니다. 또한 Worker 스레드를 관리하기 위한 Dispatcher를 생성하는 곳이기도 합니다.

SourceCache는 Transform과 Source로 구성되며, Source를 생성하고 캐싱하는 등의 역할을 맡습니다. Transform은 어떤 Source를 불러올지에 필요한 값들(줌레벨, 위도 경도, 지도가 기울어진 각도 등)을 관리하는 객체입니다. Source는 지도 렌더링에 필요한 소스들로, VectorTile, RasterTile, Image, Video 등이 있습니다.

Soucre는 WorkerSource와 짝을 이루는 객체들입니다. Source는 메인 스레드에서 이뤄지는 작업들을 메소드로 갖고 있고, WorkerSource는 Worker 스레드상에서 처리되는 메소드를 갖고 있습니다. 이를테면 VectorTileWorkerSource는 loadTile 메소드를 갖고 있는데, 이는VectorTileSource 의 loadTile 메소드에서 Actor를 통해 실행됩니다.

Tile은 주어진 벡터타일세트를 렌더링할 수있게 Bucket 과 imageAtlas 등으로 구성된 객체입니다. Tile은 WorkerTile 객체의 parsing 메소드를 통해 Worker 스레드에서 생성됩니다.

Bucket은 StyleLayer 단위로 Feature(지도에 그려질 객체)들을 변환한, WebGL이 이해할 수 있는 형태의 제일 로우한 단계의 데이터 단위입니다.

렌더링 플로우

Map 객체가 생성되면 Map은 Style과 Painter, 그리고 지도가 그려질 Container(Canvas)을 생성합니다. Style은 생성될때 style.json을 ajax로 요청하고, style에 명세된 layer들을 parse, validate합니다. 또한 style.json에는 필요한 source가 적혀있는데, Style이 갖고있는 SourceCache 에서 이 원본 source 데이터를 불러옵니다.

SourceCache는 Source가 필요한 영역을 지정하기 위한 Transform 객체를 갖고 있습니다. 이 Transform 객체를 바탕으로 style에 명시되었던 source를 ajax로 요청합니다. (source의 다운로드와 파싱은 webWorker 스레드에서 일어납니다.)

어떤 타일을 불러올지를 transform이 결정했다면, 불러온 타일에서 어떤 layer와 feature를 렌더링할지는 앞서 파싱됐던 styleLayer가 정합니다. StyleLayer는 렌더링될 순서와 필터할 feature의 속성, 렌더링 타입 등을 Tile 객체에 전달하고 Tile은 이에 맞춰 Bucket을 각 StyleLayer 별로 생산합니다.

타일이 모두 로드되고 파싱되어 렌더링할 준비가 완료되었다면 data 이벤트를 호출하고, Map 객체가 이 이벤트를 들은 후 Painter의 render 함수를 실행합니다. render 함수는 렌더링할 Tile의 layer 순서에 맞게 draw 합니다. draw는 렌더링 타입(polygon, fill, circle..) 별로 정의되어있으며, draw는 program에 layer의 bucket 데이터를 전달합니다. program은 주어진 bucket(그려야 할 좌표값들의 배열)과 shader을 이용해 그 layer를 그리는데 적합한 gl 세팅을 하고, 최종적으로 webGL api가 GPU를 이용해 렌더링하여 canvas에 결과물을 출력합니다.