衛星データ解析技術研究会
技術セミナー（応用編）

Webアプリケーションの開発技術の習得

第二回 2025/06/27

担当講師 : 田中聡至

本日のテーマ

MapLibreを用いたWebGISのフロントエンド開発

MapLibre GL JS を例に、Webフロントエンドにおけるデファクトスタンダードな地理空間情報処理を平易に体験することを目標とする。

今回触れないこと
npmのインストール方法 (npm -v でWSLやPowershellで起動できればOK!)
WSL(Ubuntu)/Power shellの動かし方
Javascriptの文法
React / Vue / Svelteのレンダリングエンジンについて
→希望や必要があれば即習のような手配を行います。

13:30-13:40 イントロ : 先週の振り返りや質問対応
13:40-14:30 MapLibreを動かすまでの設定
14:30-14:45 色々なデータが読み込めることを見てみよう
14:45-14:50 -----(休憩)-----
14:50-15:30 Leafletや他のライブラリとの比較をしてみよう
15:30-16:00 (プチアイディアソン) システムを思案しよう

MapLibre GL JSとは？

https://maplibre.org

Mapbox社が開発しているMapbox GL JSが開発していく中で、Mapboxのシステムを利用するOSS開発体制となったため、オープンプラットフォームで利用しやすく、かつ独立して開発できる体制を保つためにフォークして開発が始まったWebGLベースの地図表示ライブラリです。

WebGLベースであることを活かして、3Dの演算も行え、Globeビューなども提供しています。

ドキュメント

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

日本語版

https://unopengis.github.io/learning/ja/intro

Map Libre　GL JSで作られた面白いプロダクト

- 今ここ何番地？
- リミット１２
- MAPPLE法務局地図ビューア

上記二つは土地家屋調査士の方が作ってらっしゃいます。

今回よく使いそうな座標 (コピペ用)

```
longitude: 131.2467,  // 宇部市の経度
latitude: 33.9519,    // 宇部市の緯度
```

```
[131.2467,33.9519]
```

MapLibre　GL JSを導入しよう

二種類の読み込み方が可能です。

- npm
- CDN

npmを使って読み込む

(Node.jsをインストールしている前提で)

npm install maplibre-gl

npmを使うとなにが嬉しいか

npmを用いることで一通りライブラリをPCの中にインストールして置けるので、インターネットがない環境でもローカルにおける開発をすることができる。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>MapLibre GL JS - npm版</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
        body { margin: 0; padding: 0; }
        #map { position: absolute; top: 0; bottom: 0; width: 100%; }
    </style>
</head>
<body>
    <div id="map"></div>
    <script src="main.js"></script>
</body>
</html>
```

```
import maplibregl from 'maplibre-gl';
import 'maplibre-gl/dist/maplibre-gl.css';

const map = new maplibregl.Map({
    container: 'map',
    style: 'https://tile.openstreetmap.jp/styles/osm-bright-ja/style.json',
    center: [139.7670, 35.6814], // 東京駅
    zoom: 10
});

// ナビゲーションコントロールを追加
map.addControl(new maplibregl.NavigationControl());

// 地図の読み込み完了時の処理
map.on('load', () => {
    console.log('地図の読み込みが完了しました');
});
```

`import` キーワードを用いることでローカルに存在するライブラリを使うことができます。
npmはビルド時にこのパス解決をしてくれている格好です。
(C/C++のリンカーがこれに当たる他、Pythonであれば同じimportキーワードが存在します。)

実際に実行してみましょう

```
npm install --save-dev vite
```

```
{
    "scripts": {
        "dev": "vite",
        "build": "vite build",
        "preview": "vite preview" //こちらはどちらでも...
    }
}
```

```
npm run dev
```

- jsの名前はmain.jsやindex.jsが好まれます。
- scriptディレクトリを作成し、その中にコレクションしても良いですし、htmlと同階層においてもOK。

他には parcel や webpackによるバンドルが行われてきました。

CDNから読み込む

`<head>`タグの中で記述することでライブラリを読み込み読み込めます。

バージョン指定

```
<script src="https://unpkg.com/maplibre-gl@3.6.2/dist/maplibre-gl.js"></script>
<link href="https://unpkg.com/maplibre-gl@3.6.2/dist/maplibre-gl.css" rel="stylesheet" />
```

一番最新のものを呼びだす

```
<link href="https://unpkg.com/maplibre-gl@latest/dist/maplibre-gl.css" rel="stylesheet">
<script src="https://unpkg.com/maplibre-gl@latest/dist/maplibre-gl.js"></script>
```

```html
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="utf-8">
    <title>MapLibre GL JS - CDN版</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- MapLibre GL JS CSS -->
    <link href="https://unpkg.com/maplibre-gl@latest/dist/maplibre-gl.css" rel="stylesheet">

    <style>
        body { margin: 0; padding: 0; }
        #map { position: absolute; top: 0; bottom: 0; width: 100%; }
    </style>
</head>
<body>
    <div id="map"></div>

    <!-- MapLibre GL JS JavaScript -->
    <script src="https://unpkg.com/maplibre-gl@latest/dist/maplibre-gl.js"></script>

    <!-- 地図の初期化 -->
    <script>
        const map = new maplibregl.Map({
            container: 'map',
            style: 'https://tile.openstreetmap.jp/styles/osm-bright-ja/style.json',
            center: [139.7670, 35.6814], // 東京駅
            zoom: 10
        });

        // ナビゲーションコントロールを追加
        map.addControl(new maplibregl.NavigationControl());

        // 地図の読み込み完了時の処理
        map.on('load', function() {
            console.log('地図の読み込みが完了しました');
        });
    </script>
</body>
</html>
```

(起動の様子 / できることの解説を提示)

ユーザーインタラクション (地図に重ねるボタン)について

地図を読み込んだ後に、mapインスタンスに対して、addControlメソッドでボタンを配置することができます。

```
map.addControl(new maplibregl.NavigationControl(), 'top-right');
map.addControl(new maplibregl.ScaleControl(), 'bottom-left');
map.addControl(new maplibregl.FullscreenControl(), 'top-left');
map.addControl(new maplibregl.GeolocateControl(), 'bottom-right');
```

(このGISの要件に足る組み込みメソッドの充実さこそが一つのMapLibreの価値だと思います。)

別冊付録参照

https://alt9800.github.io/2025-RemoteSensingSeminar/handson/2025-06-27/3-userInteractions/

距離と測地系について

MapLibreでは前述のようにスケールバーの表示が行えるため、m単位で距離を測ることができます。

つまり、投影座標系(平面直角座標系)として扱われています。

具体的には **EPSG:3857** の Web Mercator (Pseudo-Mercator)です。

(Google Mapをはじめ　特にTileMapにおいて採用されている)

ただし、MapLibreにおいてはこれは経度方向(東西方向)の誤差が小さくなるように計算されているため、回転させたときに誤差が変動します。なお、円を描く際などはこれらが考慮されて描画されます。

回転角度	表示距離	実際の距離	誤差
0°	1km	1.000km	0%
30°	1km	1.155km	+15.5%
45°	1km	1.414km	+41.4%
60°	1km	2.000km	+100%
90°	1km	1.000km	無限大に発散

(直角三角形の性質より単純計算)

投影座標系と地理座標系についてはこちらを参照ください。

https://lemulus.me/column/epsg-list-gis

なお、GPSについては **EPSG：4326 (WGS84)** を利用しているため、各GISにおいては、緯度経度で与えた情報を、描画時に投影座標系に変換している格好です。

背景地図について

MapLibreでは従来の地図アプリケーションのようにタイルマップを読み込むことができる他、
ベクターレイヤーを読み込むことができます。
これらはjson形式でスタイリングを読み込まれます。

参考実装
https://alt9800.github.io/2025-RemoteSensingSeminar/handson/2025-06-27/4-backgroundMaps/

透明度の変更を行うこともできます(以下はスライダーで変更をかける例)

```
<div>
    <label for="opacity">透明度: <span id="opacity-value">100%</span></label>
    <input type="range" id="opacity" min="0" max="100" value="100">
</div>

<script>
const slider = document.getElementById('opacity');
const valueDisplay = document.getElementById('opacity-value');

slider.addEventListener('input', (e) => {
    const value = e.target.value;
    const opacity = value / 100;

    valueDisplay.textContent = value + '%';

    // 特定のレイヤーの透明度を変更
    map.setPaintProperty('my-layer-id', 'raster-opacity', opacity);
});
</script>
```

ズームレベルとタイルを表示するようにするサンプル

https://alt9800.github.io/2025-RemoteSensingSeminar/handson/2025-06-27/4-backgroundMaps/boundary/

データは自分でもつくることができる

タイルの配信やベクター形式での配信を行うことができます。

地図のスタイル作成に特化したサービスも複数あるので、「どのレベルで自作するか」を選ぶこともできます。

- MapTiler
- HERE
- Carto
- Felt

現場のプロがわかりやすく教える 位置情報デベロッパー養成講座 - 秀和システム あなたの学びをサポート！ をみるべし！

グローブビューについて

Display a globe with a fill extrusion layer - MapLibre GL JS

MapLibre にデータを読み込んでみよう！

山口県の公共データを取得してみましょう！

https://yamaguchi-opendata.jp

山口県緊急輸送道路ネットワーク計画（令和５年３月） - 緊急輸送道路（1次国道県道）.geojson - CKAN

【周南市】文化財一覧 - 【周南市】文化財一覧_csv - CKAN

csvとgeojsonの読み込みについて

https://alt9800.github.io/2025-RemoteSensingSeminar/handson/2025-06-27/6-readFiles/

距離を測ろう

福田さんからいただいていた教材作成案を実装してみました。

> ①あらかじめ、地図上にポイントが２点表示されていて、その距離が計測できる。
> ②地図上にポイントを１点打って現在地とポイント間の直線を結んで距離が計測できる。
> ③地図上に既に表示されたポイントを任意の場所に動かす。

https://alt9800.github.io/2025-RemoteSensingSeminar/handson/2025-06-27/7-distance/

その他...時間があれば触れます。

3Dの表示

点群の表示

追跡アニメーションについて

https://github.com/maplibre/maplibre-gl-directions

MapLibreの利用についてのルール / 背景地図のライセンス

MapLibreを用いて制作していることを表記しておくと丁寧かも。
背景地図について、Web媒体としては使えるが、スクリーンショットなどとして使う時の権利表記がなされていなくて使えないものもあり注意が必要です。

補足記事

npmとは？　(Node Package Manage)

JavaScriptのパッケージ管理ツールで、Node.jsのエコシステムにおいて、ライブラリのインストールや管理、共有を行うことができます。
(他の開発者と環境を揃えることに役に立つ。)

2009年にNode.jsが登場したのち、次の年にはnpmはリリースされていたようです。

CDNとは？ (Content Delivery Network)

Webコンテンツをインターネット系で配信/利用するために用意されたネットワーク網のこと。
ネットワーク網の中で負荷分散がされることによって、アクセスの集中などに強く、静的なコンテンツをアプリケーションサーバーでの処理を介さずに返すことができる。
Web開発の文脈では、CDNやJSを配信することで、html内から呼び出して、適用することができる。

類似技術 : Webサーバー(NGINXなど)、DNS

CDNのサービス

- jsDelivr
- CDNJS
- unpkg
- Cloudflare
- Akamai
- Fastly
- Microsoft Azure / Google Cloud / AWS にもそれぞれサービスがあります。

Javascriptが動く仕組み

Javascriptも機械語にコンパイルすることでコンピュータに解釈され動作します。
特に、Javacriptは「ブラウザ」によってコンパイルされ処理を返すことができることが特徴です。
Node.jsにおいてはJSをコンピュータ内が解釈できる形にして処理を返す点は他の高級言語(C、Python、Rubyなど)と同様といえます。
コンパイラとしてV8エンジンがNode.jsやChrome(Chromium)では活用されています。
(Google Apps ScriptでもV8エンジンでJavacriptを利用しているそうです。)

ランタイムについて

Javascriptの実行環境としては最近では DenoやBunのような代替が誕生しています。
DenoではTypeScriptがサポートされていたり、ライブラリの導入がかなり簡略化されていて、パッケージ管理が楽になっています。また、マルチスレッドであるため、並列なプログラミングを行うことができます。
BunはそもそもZigによって処理を高速化しているのでかなりハイパフォーマンスな実行速度を誇ります。JavaScirptの実行環境としても、バンドル速度としても、既存のランタイムと差別化できていると言えるでしょう。

npmを利用してJavascriptをサーバーサイドで動かしてみよう

おまけを参照

https://alt9800.github.io/2025-RemoteSensingSeminar/handson/2025-06-27/0-node-lessons/

TypeScriptも動かせます

```
node --experimental-strip-types app.ts
```

TSのサンプルコード

```typescript
// demo.ts
interface User {
  name: string;
  age: number;
}

// 型安全性 - 間違った型を渡すとエラーになる
function greet(user: User): string {
  return `こんにちは、${user.name}さん（${user.age}歳）！`;
}

// 自動補完とタイプセーフティ
const users: User[] = [
  { name: "田中", age: 25 },
  { name: "佐藤", age: 30 }
];

// 型推論 - TypeScriptが自動で型を判定
const messages = users.map(user => greet(user));

console.log("=== TypeScriptの良さデモ ===");
messages.forEach(msg => console.log(msg));
```

実践的には学習は`tsc`で始めて、プロダクト環境はviteを用いて開発するとよいかもしれません。

その他の実行環境について

paiza.io

wandbox

codepen

node.jsのバージョン管理について

基本的に最新版(LTS)をインストールしておくと問題はないのですが、古いバージョンでの検証がしたくなる場合は、

npm

npxコマンド

一度だけ実行したいnpmライブラリのソフトウェアがある場合はnpxが便利です

npx create-next-app my-next-app
とか

node_moduleとは

npm でインストールしたJavascriptのライブラリがモジュールとして格納される**ディレクトリ**です。

npmにはJavascriptのライブラリにとどまらずPC全体で使える便利なライブラリもあり(Gemini codeとか)、
ホームディレクトリにPC全体で使えるコマンドをインストールすることもあります。
(グローバルインストール)

(win-get , homebrew , apt に相当する使い方)

package.jsonとは

package.jsonはJSを用いた開発における設計図の役割を果たします。
依存関係の管理
スクリプトの定義
バージョン管理を行えます。

npm install --save-dev npm install --saveでインストールすると自動で記載してくれます。

これを他の開発者に渡すことで、package.jsonが存在する階層でnpm installをしてもらえば同じ環境を用意しやすいです。

package-lock.jsonではライブラリ間の依存関係や互換性のあるバージョンの担保などがされます。

ちなみに、gitの追跡はpackage.jsonとpackage-lock.jsonをしておけば、node_modules/を追跡する必要はありません。

フロントエンド事情

最近ではDockerによる開発もされているが、講師個人としてはフロントエンドでDocker開発はやや冗長かも。

また、npmに変わって、yarnやpnpmなども使われるようになってきています。

JSONとはどのような構造か

```
{
  "name": "太郎",
  "age": 25,
  "isStudent": true,
  "hobbies": ["読書", "サッカー", "旅行"],
  "address": {
    "city": "東京",
    "zipCode": "100-0001"
  }
}
```

ではGeojsonは？

```
{
  "type": "Feature",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [139.6917, 35.6895],
        [139.7661, 35.6812],
        [139.7647, 35.6126],
        [139.6917, 35.6895]
      ]
    ]
  },
  "properties": {
    "name": "東京エリア"
  }
}
```

WASM(WebAssembly)とは

最近のWebフロントにおいては、ネイティブの動作をWebに移植する目的で、Web上でJavascriptのAPIを利用してインスタンスを作成して実行するバイナリをJavascriptと連携して配置することが増えています。

→ネイティブの重い処理の移行例としてはUnityのWebGLビルドであったり、ffmpegのWeb移植の例などがあります。

Javascriptの標準化について

Javascriptにおける機能の合意などの標準化についてはEuropean Computer Manufacturers Association(ECMA)が行なっており、さまざまな実行環境での共通化に取り組んでいます。
現行のJavascriptはECMAScript 6(2015)とも呼ばれます。

`let`によるレキシカルスコープ、アロー関数を用いたthisバインディングの一つ上の階層への固定化によるコールバックの簡便化、バッククオートを用いたバインディング、クラス構文、import/exportによるモジュールの活用、Promise型の導入による非同期処理の強化、などが目玉でした。

インタラクティブとはどういうことか？

https://www.uxpin.com/studio/jp/blog-jp/魅力的でインタラクティブなウェブサイト/
実際に体験してみると良いかと思います。

他のフロントエンド地図ライブラリ事情

DeckGLについて

公式ページ

https://deck.gl

https://deck.gl/examples

さらに知見を知りたい方はKeplerGLのコードを読むと良いかもしれません。

https://kepler.gl

DeckGLの例

https://github.com/alt9800/Publications/tree/main/getting-started-deckgl

https://github.com/alt9800/sample-maps/tree/main/deck-sample

試行錯誤例

https://alt9800.github.io/Publications/getting-started-deckgl/

https://alt9800.github.io/sample-maps/deck-sample/

Cesiumの例

https://cesium.com/learn/cesiumjs-sandcastle/

Cesium Ionを用いた災害状況のリポート(2024年能登半島震災)

https://3dview.tokyo-digitaltwin.metro.tokyo.lg.jp/#share=s-zDdPb4LohEB1jylW

3DGS版も
https://noto3d.info/works/wajima_kawai_20240627

Leafletとの比較

geojsonを追加したい！

```
const map = L.map('map').setView([33.9519, 131.2467], 13);

// タイルレイヤー
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);

// GeoJSON地物の追加
const geojsonFeature = {
  type: "Feature",
  geometry: {
    type: "Polygon",
    coordinates: [[[131.24,33.95],[131.25,33.95],[131.25,33.96],[131.24,33.96],[131.24,33.95]]]
  },
  properties: {
    name: "エリアA"
  }
};

L.geoJSON(geojsonFeature, {
  style: {
    color: "#ff7800",
    weight: 2,
    fillOpacity: 0.5
  },
  onEachFeature: function (feature, layer) {
    layer.bindPopup(feature.properties.name);
```

MapLibreの例

全ての地物のスタイル指定がKey-Valueの形で統一されていることで人間がデータを読み書きしやすくなっていると思います。

```
const map = new maplibregl.Map({
  container: 'map',
  style: {
    version: 8,
    sources: {},
    layers: []
  },
  center: [131.2467, 33.9519],
  zoom: 13
});

// 地図読み込み完了後に追加
map.on('load', function () {
  map.addSource('my-geojson', {
    type: 'geojson',
    data: {
      type: "Feature",
      geometry: {
        type: "Polygon",
        coordinates: [[[131.24,33.95],[131.25,33.95],[131.25,33.96],[131.24,33.96],[131.24,33.95]]]
      },
      properties: { name: "エリアA" }
    }
  });

  map.addLayer({
    id: 'my-polygon-layer',
    type: 'fill',
    source: 'my-geojson',
    paint: {
      'fill-color': '#ff7800',
      'fill-opacity': 0.5
    }
  });

  map.addLayer({
    id: 'my-outline',
    type: 'line',
    source: 'my-geojson',
    paint: {
      'line-color': '#ff0000',
      'line-width': 2
    }
  });

  map.on('click', 'my-polygon-layer', (e) => {
    const name = e.features[0].properties.name;
    new maplibregl.Popup()
      .setLngLat(e.lngLat)
      .setText(name)
```

D3.jsの例

統計ソフトウェア

https://alt9800.github.io/spaceappchallenge2024/

似たようなことは他のデータ可視化基盤(Tableauなど)でも容易にできます。

抑えておくべきか？　Overtureマップ

https://overturemaps.org

```
# ライブラリのインストール
pip install overturemaps

# コマンドでデータをダウンロード（例: ボストン中心部の建物データ）
overturemaps download --bbox=-71.068,42.353,-71.058,42.363 -f geojson --type=building -o boston.geojson
```

コードはこれくらいではあるが...

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>OvertureMapを表示</title>
    <meta name="viewport" content="initial-scale=1,maximum-scale=1,user-scalable=no" />
    <script src='https://unpkg.com/maplibre-gl@4.1.2/dist/maplibre-gl.js'></script>
    <link href='https://unpkg.com/maplibre-gl@4.1.2/dist/maplibre-gl.css' rel='stylesheet' />
    <style>
        body { margin: 0; padding: 0; }
        #map { position: absolute; top: 0; bottom: 0; width: 100%; }
    </style>
</head>
<body>

<div id="map"></div>
<script>
    const map = new maplibregl.Map({
        container: 'map', // container id
        style: 'https://demotiles.maplibre.org/style.json', // スタイルURL
        center: [-71.06, 42.36], // 中心座標 [lng, lat]
        zoom: 15 // ズームレベル
    });

    map.on('load', () => {
        // ダウンロードしたGeoJSONを読み込む
        map.addSource('buildings', {
            'type': 'geojson',
            'data': './boston.geojson' // GeoJSONファイルへのパス
        });

        // 読み込んだデータを地図上に追加する
        map.addLayer({
            'id': 'buildings-layer',
            'type': 'fill',
            'source': 'buildings',
            'paint': {
                'fill-color': '#0080ff',
                'fill-opacity': 0.5
            }
        });
    });
</script>

</body>
</html>
```

使いやすい書籍

ハンズオンJavaScript オライリー
https://www.oreilly.co.jp/books/9784873119229/
どのような感じでJavascriptが動いているか、標準ライブラリにはどのようなものがあるか(Javascriptがどのようなことができるか)といったことがサーバーサイドJSのベテランの目線でまとまっています。

現場のプロがわかりやすく教える 位置情報エンジニア養成講座 秀和システム
https://www.shuwasystem.co.jp/book/9784798068923.html
MapLibreを活用したWebフロントについての詳細な解説がなされています。

これからWebをはじめる人のHTML＆CSS、JavaScriptのきほんのきほん マイナビブックス
https://book.mynavi.jp/ec/products/detail/id=65861
社内に広く、マークアップから動的なWebについて伝える際にあると便利な一冊で

Web知識全般

https://gihyo.jp/magazine/SD

https://gihyo.jp/magazine/SD/archive/2023/202308
上記の号から位置情報エンジニアリングのすすめの連載が始まり、現在ではMIERUNE社のブログとして公開されています。
https://zenn.dev/mierune_inc/books/location-engineering

Web Designing 2022年12月号 | Web Designing Web
フロントエンドは今時どうすべきかが学べます。
これに関しては記事として公式に転載されています。
https://webdesigning.book.mynavi.jp/article/4876/

(この辺りは呼んでくだされば各社個別で講習も行います。)

Javascriptに関する情報を集めるには

https://ics.media

web技術全般

https://www.publickey1.jp

これからJSを学ぶなら...

AjaxとjQueryは避けた方がいいかも

外部との接続について

現代のmarkupでは概ね fetchAPIが用いられます。
thenメソッドチェーンはとっつきにくく感じると思いますが、最初はお作法として覚えてもいいかもしれません。

[非同期処理の仲間たち]
Async / Await
XMLHttpRequest
Ajax
Axios
fetch

gitの使い方についての補足

空の「ディレクトリ」は管理対象に取れないことに注意

今週のジオニュース

帝国書院さんのジオグラフ (2021)がにわかに盛り上がっている

https://www.geograph.teikokushoin.co.jp

データソースに関する展望

利用できる地物が増えつつあります。
道路情報に関しては国土交通省がXRoadプロジェクトを立ち上げているので、こちらに意見を投げてみましょう。

DEMも拡充されつつあります。

その他のニュース

彗星の如く現れたGemini CLI
https://cloud.google.com/blog/ja/topics/developers-practitioners/introducing-gemini-cli/

ちなみにClaude Codeも

```
npm install -g @anthropic-ai/claude-code
```

でインストールできる

アンケート

Markup(html / css / Javascript)の開発経験について教えてください。
また、同時に、普段、どのような言語での開発業務に携わっているかを記載ください。

本年度、および来年度の解説の指標になります。

JSONがわかる (任意のプログラムでJSON形式を表現あるいは読み込みと生成ができるか)

WebAPI (REST)がわかる