

# FOSS4G 2025 KANSAI ハンズオンデイ

## すこし"Deep"なLeaflet エコシステム

### インターラクティブな地図アプリケーション開発

## 今回の趣旨：

Leafletでできることをいろいろみながら、時折手を動かしつつ、実装眺めて、  
LLMから吐かれるコードを理解できるようになろう！！

(座学が多いのでみんなで和気藹々できればいいな / 実装したい！ってところがあれば  
隨時止めてください。)

※：みなさんの様子をみながら講義内容をアレンジしていきます。

機能要件や設計を理解して、安定したコードを吐かせれるようになろう

## サイト

<https://www.osgeo.jp/events/2025-2/foss4g-2025-kansai/foss4g-2025-kansai-ハンズオンデイ#Deep>

## 資料

<https://alt9800.github.io/2025-RemoteSensingSeminar/>

## ソースコード

<https://github.com/alt9800/2025-RemoteSensingSeminar/tree/main/2025-07-06>

## 元ネタ

2024年度 衛星データ解析技術研究会 技術セミナー（応用編）講義内容

ソースコード

<https://github.com/alt9800/sample-maps/tree/main/Seminar>

スライド

<https://drive.google.com/drive/folders/1mpv-9r0SVznrseM16jQox6dWk6aayStv?usp=sharing>

# 自己紹介

講師紹介：田中

✉ : [alt9800jp@gmail.com](mailto:alt9800jp@gmail.com)

OSGeoJP 運営委員

最近やってること

LiDARや機械学習による測量支援 / 屋内の地図化ツール/  
点字ブロック Lineデータ作成 /Web における3D

お仕事

<https://ikominaprj.xsrv.jp/AR/>

趣味

<https://simplespeedtest-amaranth.web.app>

<https://alt9800.github.io/visualization/prefecture-viewer/>

登壇

iOSDC2025 : <https://fortee.jp/iosdc-japan-2024/proposal/afbcd097-0da9-4073-8f48-528f007e28b7>



最近つくってるもの

<https://solemate-3xn.pages.dev>

# 基礎についてのアジェンダ

1. Leaflet基礎 - 地図の表示と基本操作
2. インタラクティブ機能 - ドラッグ&ドロップ、メニューバー実装
3. Firebase連携 - リアルタイムCRUDシステム
4. 高度な可視化 - ヒートマップと地図分割表示
5. パフォーマンス最適化 - 大量データの効率的な描画

# 第1章

## Leaflet基礎編

# Leafletとは？

- 軽量でモバイルフレンドリーなJavaScript地図ライブラリ
- オープンソースで豊富なプラグイン
- 主要なブラウザで動作

```
// 基本的な地図の初期化
var map = L.map('map').setView([34.180873, 131.469966], 8);
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '© OpenStreetMap contributors'
}).addTo(map);
```

# 実装例1: 背景地図の切り替え

```
// OpenStreetMapレイヤー
var osmLayer = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png');

// 航空写真レイヤー
var satelliteLayer = L.tileLayer(
  'https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}'
);

// レイヤーコントロールの追加
L.control.layers({
  "OpenStreetMap": osmLayer,
  "航空写真": satelliteLayer
}).addTo(map);
```

## 学びのポイント：タイルレイヤー

- タイルサーバーの選択
  - OpenStreetMap（無料、制限あり）
  - 地理院タイル（日本向け）
  - 商用サービス（Mapbox、Google Maps）
- パフォーマンスの考慮
  - タイルのキャッシュ
  - 適切なズームレベルの設定

## 第2章

### インタラクティブ機能の実装

## 実装例2: GeoJSON ドラッグ&ドロップ

```
map.getContainer().addEventListener('drop', function(e) {
  e.preventDefault();
  var file = e.dataTransfer.files[0];
  var reader = new FileReader();

  reader.onload = function(event) {
    var geoJsonData = JSON.parse(event.target.result);
    L.geoJSON(geoJsonData, {
      pointToLayer: function(feature, latlng) {
        return L.circleMarker(latlng, {
          radius: 8,
          fillColor: "#ff7800"
        });
      }
    }).addTo(map);
  };
  reader.readAsText(file);
}) alt9800
```

## 実装例3: インタラクティブなメニューバー

```
// ズームレベル制御
function changeZoom(value) {
    map.setZoom(value);
    document.getElementById('zoomLevel').textContent =
        'ズームレベル: ' + value;
}

// マーカーの表示/非表示
function toggleMarkers() {
    if (markersVisible) {
        map.removeLayer(markers);
    } else {
        map.addLayer(markers);
    }
    markersVisible = !markersVisible;
}
```

# 学びのポイント：ユーザビリティ

- 直感的な操作
  - ドラッグ&ドロップでファイル読み込み
  - スライダーでズーム調整
- 視覚的フィードバック
  - ホバー効果
  - アニメーション
- レスポンシブデザイン
  - モバイル対応
  - タッチ操作のサポート

## 第3章

### Firebase連携とCRUD操作

## 実装例4: Firestoreとの連携

```
// Firebaseの初期化
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

// マーカーの追加
const newMarker = {
  Comment: comment,
  Location: new GeoPoint(lat, lng),
  CreatedAt: serverTimestamp(),
  userName: userName
};

await setDoc(doc(db, 'LatLngs', mapName, 'markers', id), newMarker);
```

# リアルタイムデータ同期

```
// マーカーの読み込みと表示
async function loadMarkers() {
    const querySnapshot = await getDocs(
        collection(db, 'LatLngs', selectedMap, 'markers')
    );

    querySnapshot.forEach((doc) => {
        const data = doc.data();
        const marker = L.marker([
            data.Location.latitude,
            data.Location.longitude
        ]).addTo(map)
            .bindPopup(`<b>${data.userName}</b><br>${data.Comment}`);
    });
}
```

# 学びのポイント：データ管理

- NoSQLデータベースの活用
  - GeoPointでの位置情報管理
  - タイムスタンプによる履歴管理
- CRUD操作の実装
  - Create: 新規マーカー追加
  - Read: マーカー一覧表示
  - Update: コメント更新
  - Delete: マーカー削除

## 第4章

### 高度な可視化技術

## 実装例5: ヒートマップ表現

### Leaflet.heat

```
var heatData = [
  [34.180873, 131.469966, 0.5],
  [33.968776, 130.940482, 0.5],
  // ...
];

L.heatLayer(heatData, {
  radius: 50,
  gradient: {0.4: 'blue', 0.65: 'lime', 1: 'red'}
}).addTo(map);
```

## 複数の実装方法比較

ライブラリ	特徴	用途
<b>Leaflet.heat</b>	軽量、簡単	基本的なヒートマップ
<b>MapLibre GL</b>	GPU活用、高速	大量データ
<b>Deck.gl</b>	3D対応、高機能	高度な可視化

## 実装例6: Side-by-Side地図比較

```
// 2つのレイヤーを左右に分割表示
L.control.sideBySide(gsiLayer, osmLayer).addTo(map);

// カスタムディバイダーの実装
_onDividerDrag: function(e) {
    var rect = this._container.getBoundingClientRect();
    var percent = (e.clientX - rect.left) / rect.width;
    this._updateClip();
}
```

## 学びのポイント：UI/UX設計

- 比較機能の実装
  - スワイプ操作で直感的な比較
  - 同期スクロール
- カスタムコントロール
  - 独自UIコンポーネントの作成
  - イベントハンドリング

## 第5章

### パフォーマンス最適化

# 大量データの効率的な描画

```
// GeoJSONデータの最適化
L.geoJSON(data, {
  style: function(feature) {
    return {
      fillColor: getColor(feature.properties.value),
      weight: 1, // 細い境界線
      fillOpacity: 0.7
    };
  },
  // シンプルなポップアップ
  onEachFeature: function(feature, layer) {
    layer.bindPopup(feature.properties.name);
  }
}).addTo(map);
```

# パフォーマンス改善テクニック

## 1. データの軽量化

- 座標精度の調整
- 不要なプロパティの削除

## 2. レンダリング最適化

- Canvas vs SVG の選択
- クラスタリングの活用

## 3. 遅延読み込み

- ビューポート内のみ表示
- ズームレベルに応じた詳細度

## 実装例7: マーカークラスタリング

```
// 多数のマーカーをクラスタリング
var markers = L.markerClusterGroup({
  chunkedLoading: true,
  spiderfyOnMaxZoom: true
});

// マーカーを追加
data.forEach(function(item) {
  var marker = L.marker([item.lat, item.lng]);
  markers.addLayer(marker);
});

map.addLayer(markers);
```

## まとめと次のステップ

# 学んだこと

## ✓ 基礎技術

- Leafletの基本操作とレイヤー管理

## ✓ インタラクティブ機能

- ドラッグ&ドロップ、動的UI

## ✓ データ連携

- Firebase/Firestoreとのリアルタイム同期

## ✓ 高度な表現

- ヒートマップ、地図比較

## ✓ 最適化

2025/07/06 alt9800

# 次のステップ

## 1. プラグインの活用

- Leaflet.draw (図形描画)
- Leaflet.routing (経路検索)

## 2. モバイル最適化

- タッチジェスチャー
- オフライン対応

## 3. セキュリティ強化

- Firebase認証
- データ検証

# リソースとドキュメント

- 公式ドキュメント
  - [Leaflet公式](#)
  - [Firebase Docs](#)
- サンプルコード
  - GitHub: [/Seminar/](#)
  - [CodePen examples](#)
- コミュニティ
  - [OSGeo\(JP\)](#)
  - [OpenStreetMap Foundation Japan](#)

さらに面白そうな使い方

<https://game8.jp/pokemon-legends/424101>

タイルの読み込みをするアプリケーションなので、Staticな画像を使うことができる  
関連技術：ジオリファレンス

## QGIS to Web

<https://plugins.qgis.org/plugins/qgis2web/>

# サンプル

<https://alt9800.github.io/sample-maps/sukesan-2023-07-29/#8/34.441/132.004>

## デフォルトの座標系

Leafletでは、地理座標系（緯度・経度）をそのまま使用することが基本設計となっています。`L.LatLng(緯度, 経度)`という形式で座標を指定し、内部的にWeb Mercator (EPSG:3857) に投影変換して表示します。

MapLibreは、内部的にWeb Mercatorで動作しますが、APIレベルでは[経度, 緯度]の順番で配列として座標を扱います。GeoJSON仕様に準拠した設計となっているため、座標の順序がLeafletとは逆になっています

## 投影法のサポート

Leafletは、プラグイン（Proj4Leaflet）を使用することで様々な投影法をサポートできます。これにより、国土地理院の平面直角座標系など、特殊な座標系も扱えます：

javascript// Leafletでカスタム投影法を使用

```
var crs = new L.Proj.CRS('EPSG:2451',
'+proj=tmerc +lat_0=33 +lon_0=131 +k=0.9999 +x_0=0 +y_0=0 +ellps=GRS80
+units=m +no_defs');
```

MapLibreは、基本的にWeb Mercatorに最適化されており、他の投影法への対応は限定的です。カスタム投影法を使用する場合は、事前に座標変換を行う必要があります。

Leafletでは、`project()`と`unproject()`メソッドで地理座標とピクセル座標の相互変換が簡単に行えます。また、異なるズームレベルでの変換も柔軟に対応できます。

MapLibreでは、`project()`と`unproject()`メソッドも提供されていますが、主にマップボックスのベクトルタイル仕様に最適化された実装となっています。

## Leafletのとても良い使い方

クラスター表示を行う (これもMapLibreだと少し楽かも)

[Leaflet.markercluster | Marker Clustering plugin for Leaflet](#)

新党・チームみらい【公式】 @team\_mirai\_jp

[https://x.com/team\\_mirai\\_jp/status/1942617325930439140](https://x.com/team_mirai_jp/status/1942617325930439140)

ポスターマップ、実際に操作してみた画面の動画はこちら

地図上で掲示板の場所を選んで、ポスターを貼ったことを報告していただくと、  
画面上の灰色のマークが緑色に変わります <https://t.co/dh26HWN9ze>#チームみ  
らい #みらいを選ぼう [pic.twitter.com/Z7JB2tuiA9](https://pic.twitter.com/Z7JB2tuiA9)

— 新党・チームみらい【公式】 (@team\_mirai\_jp) July 8, 2025