

Leaflet.js 学習ガイドブック

～ Web 地図アプリケーション開発の基礎から応用まで ～

河合勝彦

名古屋市立大学大学院経済学研究科

kkawai@econ.nagoya-cu.ac.jp

2026 年 1 月 6 日

目次

第 1 章	Leaflet.js とは	5
1.1	概要	5
1.2	基本構成	5
1.3	本書の構成	6
1.4	開発環境について	6
第 1 部 初級編		9
第 2 章	基本的な地図表示	11
2.1	学習内容	11
2.2	HTML テンプレート	12
2.3	コード解説	13
2.4	練習問題	14
第 3 章	マーカーを追加	15
3.1	学習内容	15
3.2	コード解説	15
3.3	練習問題	15
第 4 章	ポップアップを表示	17
4.1	学習内容	17
4.2	コード解説	18
4.3	練習問題	18
第 5 章	複数のマーカー	19
5.1	学習内容	19
5.2	コード解説	19
5.3	練習問題	20
第 6 章	円を描画	23
6.1	学習内容	23
6.2	コード解説	23
6.3	練習問題	24

第 7 章	ポリゴンを描画	27
7.1	学習内容	27
7.2	コード解説	27
7.3	練習問題	28
第 8 章	ポリラインを描画	31
8.1	学習内容	31
8.2	コード解説	32
8.3	練習問題	32
第 9 章	地図のズームとパン操作	35
9.1	学習内容	35
9.2	コード解説	36
9.3	練習問題	36
第 10 章	異なるタイルレイヤー	39
10.1	学習内容	39
10.2	主なタイルプロバイダー	39
10.3	コード解説	40
10.4	練習問題	40
第 11 章	地図のイベント処理	43
11.1	学習内容	43
11.2	コード解説	44
11.3	練習問題	44
第 II 部	中級編	47
第 12 章	カスタムマーカーアイコン	49
12.1	学習内容	49
12.2	コード解説	50
12.3	練習問題	50
第 13 章	ドラッグ可能なマーカー	53
13.1	学習内容	53
13.2	コード解説	54
13.3	練習問題	54
第 14 章	GeoJSON の基本	57
14.1	学習内容	57
14.2	コード解説	58
14.3	練習問題	59

第 15 章	GeoJSON のスタイリング	61
15.1	学習内容	61
15.2	コード解説	62
15.3	練習問題	62
第 16 章	レイヤーグループ	65
16.1	学習内容	65
16.2	コード解説	66
16.3	練習問題	66
第 17 章	レイヤーコントロール	69
17.1	学習内容	69
17.2	コード解説	70
17.3	練習問題	70
第 18 章	スケールコントロール	73
18.1	学習内容	73
18.2	コード解説	74
18.3	練習問題	74
第 19 章	画像オーバーレイ	77
19.1	学習内容	77
19.2	コード解説	77
19.3	練習問題	78
第 20 章	現在地表示 (Geolocation)	81
20.1	学習内容	81
20.2	コード解説	82
20.3	練習問題	83
第 21 章	ツールチップ	85
21.1	学習内容	85
21.2	コード解説	86
21.3	練習問題	86
第 III 部	上級編	89
第 22 章	コロプレスマップ	91
22.1	学習内容	91
22.2	主なテクニック	91
22.3	練習問題	92

第 23 章 カスタムコントロール	95
23.1 学習内容	95
23.2 コード解説	95
23.3 練習問題	96
第 24 章 マーカークラスタリング（概念）	99
24.1 学習内容	99
24.2 クラスタリングの基本概念	99
24.3 練習問題	100
第 25 章 地図の境界制限	103
25.1 学習内容	103
25.2 コード解説	104
25.3 練習問題	104
第 26 章 動的なマーカー追加・削除	107
26.1 学習内容	107
26.2 コード解説	108
26.3 練習問題	108
第 27 章 アニメーション付きマーカー	111
27.1 学習内容	111
27.2 コード解説	112
27.3 練習問題	112
第 28 章 WMS レイヤー	115
28.1 学習内容	115
28.2 練習問題	116
第 29 章 非地理的マップ	119
29.1 学習内容	119
29.2 コード解説	119
29.3 練習問題	120
第 30 章 マップペイン	123
30.1 学習内容	123
30.2 コード解説	124
30.3 練習問題	124
第 31 章 総合アプリケーション（店舗検索）	127
31.1 学習内容	127
31.2 アプリケーション構成	128
31.3 コード解説	128

31.4	練習問題	129
付録 A	よく使うオプション一覧	131
A.1	L.map() オプション	131
A.2	L.tileLayer() オプション	131
A.3	パスオプション（共通）	131
付録 B	主要なタイルプロバイダー	133
B.1	タイル URL 一覧	133
B.2	Attribution（著作権表示）テンプレート	133
付録 C	参考文献・参考サイト	135
C.1	公式ドキュメント	135
C.2	地図データ・タイルプロバイダー	135
C.3	GeoJSON・地理データ	136
C.4	Leaflet プラグイン	136
C.5	Web 技術・標準仕様	136
C.6	書籍・文献	137
C.7	チュートリアル・学習リソース	137
付録 D	LLM で学ぶプロンプトセット（20 本）	139
D.1	難易度の目安	139
D.2	LLM の使い方（失敗しにくい手順）	139
D.3	品質ブースター（任意）	140
D.4	P01（★1）最小構成で地図を表示	140
D.5	P02（★1）配列からマーカー生成+ポップアップ+ツールチップ	141
D.6	P03（★2）クリックで座標取得&コピー	142
D.7	P04（★2）Polyline/Polygon と hover ハイライト	142
D.8	P05（★2）レイヤ切替（Base/Overlay）+スケール+ズーム ON/OFF	143
D.9	P06（★3）クリックで選択状態（強調・解除）+サイドパネル	143
D.10	P07（★3）複数選択（Shift+クリック）&選択リスト & GeoJSON 出力	144
D.11	P08（★3）ポップアップ内ボタン→外部処理（安全な書き方）	144
D.12	P09（★3）右クリックのコンテキストメニュー	145
D.13	P10（★3）現在地ボタン（Geolocation）+誤差円	145
D.14	P11（★3）fetch で GeoJSON 読み込み（ローディング&エラー付き）	146
D.15	P12（★3）CSV → マーカー（バリデーション込み）	147
D.16	P13（★4）コロプレス+凡例+情報ボックス（テーマ地図の王道）	147
D.17	P14（★4）地図+表の連動（双方向）	148
D.18	P15（★4）MarkerCluster（大量点の定番）	148
D.19	P16（★4）Leaflet.draw（作図→編集→GeoJSON 出力）	149
D.20	P17（★5）表示範囲内だけ取得（debounce + Abort + 差分更新）	149

D.21	P18 (★ 5) パフォーマンス比較の実験台 (Marker vs CircleMarker vs Canvas) . . .	150
D.22	P19 (★ 4) Vite + TypeScript で Leaflet (教材配布・実務の入口)	150
D.23	P20 (★ 3) デバッグ&レビュー (詰まつたらこれ)	151
D.24	おすすめの進め方	152

第1章

Leaflet.js とは

1.1 概要

Leaflet.js は、オープンソースの JavaScript ライブラリで、インタラクティブな地図を Web ページに埋め込むために使用されます。軽量（約 42KB）で高速、かつモバイルフレンドリーな設計が特徴です。

2011 年に初版がリリースされました。本書のサンプルはすべて **Leaflet 1.9.4** で動作確認しています。

1.1.1 主な特徴

- **軽量**：圧縮後約 42KB と非常に小さい
- **シンプルな API**：直感的で学習しやすい
- **モバイル対応**：タッチ操作やジェスチャーをサポート
- **プラグイン豊富**：多くの拡張プラグインが利用可能
- **オープンソース**：無料で商用利用可能

1.1.2 動作要件

- モダンブラウザ（Chrome, Firefox, Safari, Edge）
- Internet Explorer 11 以上（レガシー対応時）
- タイルサーバーへのアクセス（OpenStreetMap 等）

1.2 基本構成

Leaflet.js を使用するには、以下の 2 つのファイルを読み込みます：

```
1 <!-- CSS -->
2 <link rel="stylesheet"
3     href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
4
5 <!-- JavaScript -->
```

```
6 <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js">
7 </script>
```

Listing 1.1 Leaflet の読み込み

1.3 本書の構成

本書は 30 のサンプルを通じて、段階的に Leaflet.js を学習できる構成になっています。第 1 章は導入、第 2 章～第 31 章が 30 個のサンプル（サンプル 01～30）に対応しています。

第 I 部：初級編（第 2～11 章 / サンプル 01～10）

基本的な地図表示、マーカー、ポップアップ、図形描画、イベント処理

第 II 部：中級編（第 12～21 章 / サンプル 11～20）

カスタムアイコン、GeoJSON、レイヤー制御、画像オーバーレイ、Geolocation

第 III 部：上級編（第 22～31 章 / サンプル 21～30）

コロプレスマップ、カスタムコントロール、WMS、マップペイン、総合アプリ

注意

本書のサンプルファイル名（例：01-basic-map.html）と章番号の対応は以下の通りです：

- サンプル 01～10 → 第 2～11 章（初級編）
- サンプル 11～20 → 第 12～21 章（中級編）
- サンプル 21～30 → 第 22～31 章（上級編）

LLM を使った学習

ChatGPT や Claude などの LLM を活用して Leaflet.js を効率的に学びたい場合は、巻末の「LLM で学ぶプロンプトセット（20 本）」を参照してください。目的別のおすすめルートも用意しています：

- 最短ルート（授業 1～2 回向け）
- 実務寄り（ダッシュボード/研究）
- 作図や調査支援を作る

1.4 開発環境について

1.4.1 ローカルサーバーの使用

初級編（基本的な地図表示、マーカー、図形描画など）は、HTML ファイルをブラウザで直接開くだけで動作します。しかし、中級編以降（GeoJSON、外部データ読み込み、画像オーバーレイなど）では、セキュリティ制約により file:// プロトコルでは正常に動作しない場合があります。

以下のローカルサーバーを使用することを推奨します：

VS Code Live Server

Visual Studio Code の拡張機能。ファイル保存時に自動リロードも可能。

Python http.server

`python3 -m http.server 8000` でサーバー起動。`http://localhost:8000` でアクセス。

Node.js http-server

`npx http-server` でサーバー起動。

1.4.2 サンプルコードの配布

本書のサンプルコードは以下の GitHub リポジトリで公開しています：

- リポジトリ：<https://github.com/kkawailab/kklab-leafletJS-samples>
- `samples/` ディレクトリに全 30 サンプルの HTML ファイルを収録

第Ⅰ部

初級編

第2章

基本的な地図表示

2.1 学習内容

Leaflet.js を使って最も基本的な地図を表示する方法を学びます。

01 - 基本的な地図表示

学習内容: Leaflet.js を使って最も基本的な地図を表示します。

ポイント:

- L.map() で地図オブジェクトを作成
- setView() で中心座標とズームレベルを設定
- L.tileLayer() でタイルレイヤー（地図画像）を追加

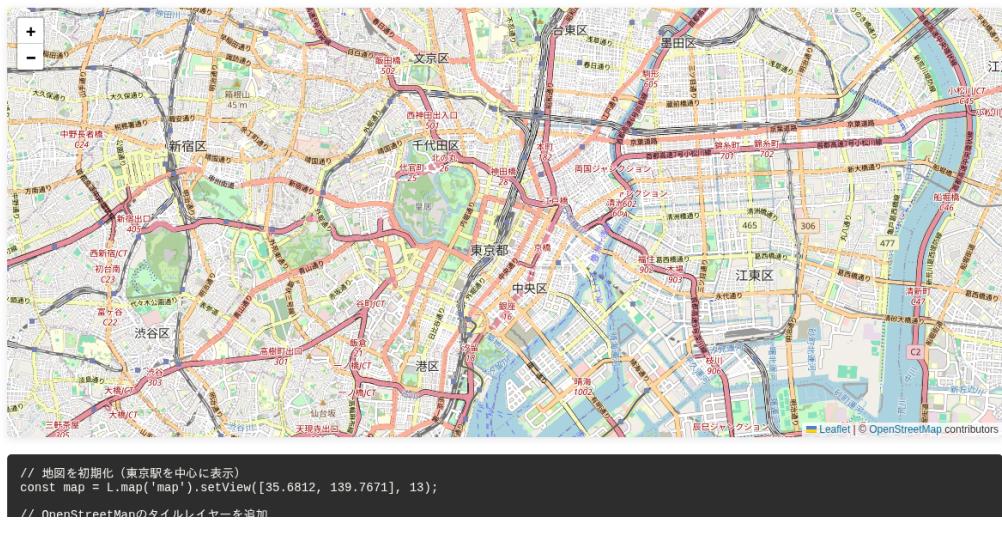


図 2.1 サンプル実行結果

ポイント

- L.map() で地図オブジェクトを作成
- setView() で中心座標とズームレベルを設定
- L.tileLayer() でタイルレイヤー（地図画像）を追加

新しいメソッド・関数

```
L.map(id) 指定した HTML 要素に地図を作成  
.setView([lat, lng], zoom) 地図の中心座標とズームレベルを設定  
L.tileLayer(url, options) タイルレイヤーを作成  
.addTo(map) レイヤーを地図に追加
```

2.2 HTML テンプレート

Leaflet.js で地図を表示するための最小構成 HTML を以下に示します。本書の全サンプルは、このテンプレートをベースにしています。

PDF からのコピー＆ペーストについて

PDF からコードをコピー＆ペーストすると、環境によっては不要な空白が混入し、動作しないことがあります。必ず GitHub 掲載のサンプルファイルをコピーしてください：
<https://github.com/kkawailab/kklab-leafletJS-samples>

```
1 <!DOCTYPE html>  
2 <html lang="ja">  
3 <head>  
4     <meta charset="UTF-8">  
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6     <title>Leaflet Map</title>  
7  
8     <!-- Leaflet CSS (必須) -->  
9     <link rel="stylesheet"  
10        href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />  
11  
12     <style>  
13         /* 重要: html/body/#map すべてに高さを指定 */  
14         html, body, #map {  
15             height: 100%;  
16             margin: 0;  
17             padding: 0;  
18         }  
19     </style>  
20 </head>  
21 <body>  
22     <!-- 地図を表示するコンテナ -->  
23     <div id="map"></div>  
24  
25     <!-- Leaflet JavaScript (必須) -->  
26     <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js">  
27     </script>  
28
```

```
29 <script>
30     // ここに地図のコードを記述
31     const map = L.map('map').setView([35.6812, 139.7671], 13);
32     L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
33         attribution: '&copy; OpenStreetMap contributors'
34     }).addTo(map);
35 </script>
36 </body>
37 </html>
```

Listing 2.1 最小構成 HTML テンプレート

地図が表示されない場合のチェックリスト

1. **高さの指定** : #map に高さ (height) が設定されていますか？
高さが 0 だと地図は表示されません。
2. **ID の一致** : HTML の <div id="map"> と JavaScript の L.map('map') が一致していますか？
3. **読み込み順序** : Leaflet.js は地図コンテナ (<div id="map">) の後に読み込んでいますか？
4. **コンソールエラー** : ブラウザの開発者ツール (F12) でエラーが出ていますか？

2.3 コード解説

```
1 // 地図オブジェクトを作成
2 // 'map' はHTML要素の id 属性
3 const map = L.map('map').setView([35.6812, 139.7671], 13);
4
5 // タイルレイヤーを追加
6 // {s}: サブドメイン, {z}: ズーム, {x}, {y}: タイル座標
7 L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
8     attribution: '&copy; OpenStreetMap contributors'
9 }).addTo(map);
```

Listing 2.2 基本的な地図表示

注意

attribution (著作権表示) は必須です。タイルプロバイダーの利用規約に従って正しく設定してください。

2.4 練習問題

練習問題

1. 大阪駅（緯度: 34.7024、経度: 135.4959）を中心に、ズームレベル 12 で地図を表示するコードを書いてください。
2. ズームレベルを 5 に設定して、日本全体が見えるように表示してください。初期中心は日本の中心付近（緯度: 36.0、経度: 138.0）にしてください。
3. CartoDB Positron(URL: https://s.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png) をタイルレイヤーとして使用する地図を作成してください。

模範解答

問題 1 の解答:

```
1 const map = L.map('map').setView([34.7024, 135.4959], 12);
2 L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}.png', {
3     attribution: '&copy; OpenStreetMap contributors'
4 }).addTo(map);
```

問題 2 の解答:

```
1 const map = L.map('map').setView([36.0, 138.0], 5);
2 L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}.png', {
3     attribution: '&copy; OpenStreetMap contributors'
4 }).addTo(map);
```

問題 3 の解答:

```
1 const map = L.map('map').setView([35.6812, 139.7671], 13);
2 L.tileLayer('https://s.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}
3 .png', {
4     attribution: '&copy; <a href="https://www.openstreetmap.org/
copyright">OpenStreetMap</a> contributors &copy; <a href="https
://carto.com/attribution">CARTO</a>'
```

第3章

マーカーを追加

3.1 学習内容

地図上に位置を示すマーカーを追加する方法を学びます。

新しいメソッド・関数

```
L.marker([lat, lng]) 指定座標にマーカーを作成
.addTo(map) マーカーを地図に追加
```

3.2 コード解説

```
1 // マーカーを作成して地図に追加
2 // 座標は [緯度, 経度] の順番
3 const marker = L.marker([35.6812, 139.7671]).addTo(map);
```

Listing 3.1 マーカーの追加

ポイント

座標は「緯度（南北）, 経度（東西）」の順番です。Google Maps のリンクなどでは逆の場合があるので注意してください。

3.3 練習問題

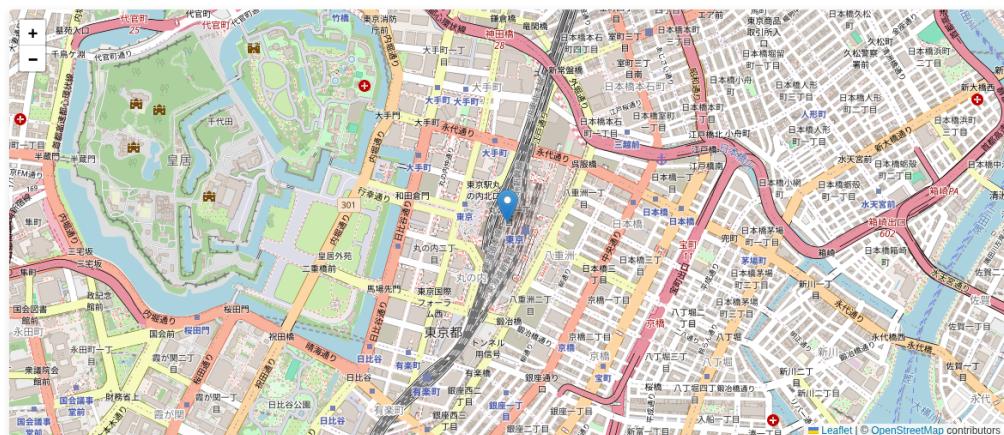
練習問題

1. 富士山の山頂（緯度: 35.3606、経度: 138.7274）にマーカーを配置してください。
2. 東京スカイツリー（緯度: 35.7101、経度: 139.8107）と東京タワー（緯度: 35.6586、経度: 139.7454）の2つのマーカーを同じ地図上に配置してください。
3. マーカーを変数に保存して、後からコンソールに座標を表示してください。

02 - マーカーを追加

学習内容: 地図上に位置を示すマーカーを追加します。

- `L.marker()` でマーカーを作成
- `addTo(map)` でマーカーを地図に追加
- 座標は「緯度、経度」の配列で指定



```
// マーカーを作成して地図に追加
const marker = L.marker([35.6812, 139.7671]).addTo(map);
```

図3.1 サンプル実行結果

模範解答

問題1の解答:

```
1 L.marker([35.3606, 138.7274]).addTo(map);
```

問題2の解答:

```
1 L.marker([35.7101, 139.8107]).addTo(map); // スカイツリー
2 L.marker([35.6586, 139.7454]).addTo(map); // 東京タワー
```

問題3の解答:

```
1 const marker = L.marker([35.6812, 139.7671]).addTo(map);
2 const position = marker.getLatLng();
3 console.log('緯度:', position.lat, '経度:', position.lng);
```

第4章

ポップアップを表示

4.1 学習内容

マーカーや地図上にポップアップ（吹き出し）を表示する方法を学びます。

03 - ポップアップを表示

学習内容: マーカーや地図上にポップアップ（吹き出し）を表示します。
ポイント:
 • bindPopup() でマーカーにポップアップをバインド
 • openPopup() で自動的にポップアップを開く
 • ポップアップ内にはHTMLを記述可能

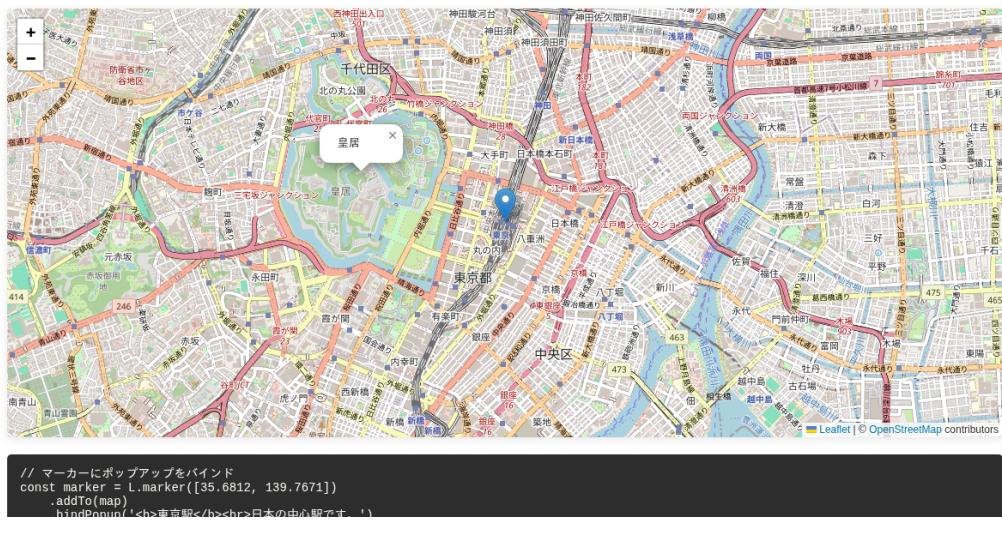


図 4.1 ポップアップの表示

新しいメソッド・関数

- .bindPopup(content) マーカーにポップアップをバインド
- .openPopup() ポップアップを開く
- L.popup() 独立したポップアップを作成
- .setLatLng([lat, lng]) ポップアップの位置を設定
- .setContent(html) ポップアップの内容を設定
- .openOn(map) 地図上にポップアップを開く

4.2 コード解説

```
1 // マーカーにポップアップをバインド
2 const marker = L.marker([35.6812, 139.7671])
3   .addTo(map)
4   .bindPopup('<b>東京駅</b><br>日本の中心駅です。')
5   .openPopup();
6
7 // 独立したポップアップ
8 L.popup()
9   .setLatLng([35.6852, 139.7528])
10  .setContent('皇居')
11  .openOn(map);
```

Listing 4.1 ポップアップの表示

4.3 練習問題

練習問題

- マーカーに「渋谷駅」というタイトルと「若者の街」という説明を含む HTML ポップアップを表示してください。
- ポップアップを最初から開いた状態にしてください (`openPopup()` を使用)。
- マーカーなしで、地図上の任意の位置に独立したポップアップを表示してください。

模範解答

問題 1 の解答:

```
1 L.marker([35.6580, 139.7016])
2   .addTo(map)
3   .bindPopup('<b>渋谷駅</b><br>若者の街');
```

問題 2 の解答:

```
1 L.marker([35.6580, 139.7016])
2   .addTo(map)
3   .bindPopup('<b>渋谷駅</b><br>若者の街')
4   .openPopup();
```

問題 3 の解答:

```
1 L.popup()
2   .setLatLng([35.6812, 139.7671])
3   .setContent('ここは東京駅付近です')
4   .openOn(map);
```

第5章

複数のマーカー

5.1 学習内容

配列データから複数のマーカーを効率的に地図上に配置する方法を学びます。

04 - 複数のマーカー

学習内容: 配列データから複数のマーカーを効率的に地図上に配置します。
ポイント:
 • 位置情報を配列で管理
 • `forEach()` でループしてマーカーを追加
 • 各マーカーに異なるポップアップを設定

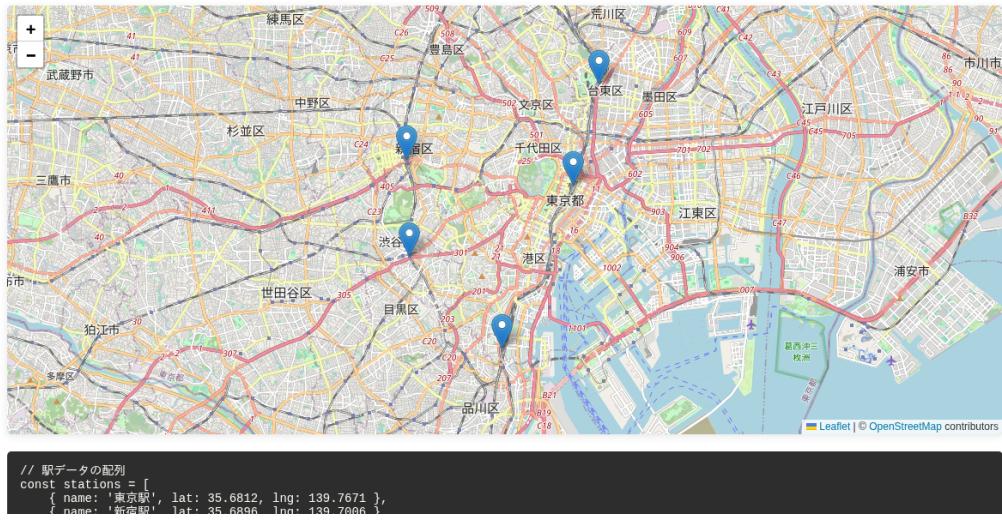


図 5.1 複数マーカーの表示

ポイント

- 位置情報を配列で管理
- `forEach()` でループしてマーカーを追加
- 各マーカーに異なるポップアップを設定

5.2 コード解説

```

1 // 駅データの配列
2 const stations = [
3     { name: '東京駅', lat: 35.6812, lng: 139.7671 },
4     { name: '新宿駅', lat: 35.6896, lng: 139.7006 },
5     { name: '渋谷駅', lat: 35.6580, lng: 139.7016 }
];
7
8 // 配列をループしてマーカーを追加
9 stations.forEach(station => {
10     L.marker([station.lat, station.lng])
11         .addTo(map)
12         .bindPopup(`<b>${station.name}</b>`);
13 });

```

Listing 5.1 複数マーカーの追加

5.3 練習問題

練習問題

1. 以下の観光地データを使って、複数のマーカーを配置してください：
 - 金閣寺（緯度: 35.0394、経度: 135.7292）
 - 清水寺（緯度: 34.9949、経度: 135.7850）
 - 伏見稻荷（緯度: 34.9671、経度: 135.7727）
2. 各マーカーに観光地名をポップアップで表示してください。
3. マーカーにインデックス番号（1, 2, 3...）を付けてポップアップに表示してください。

模範解答

問題 1~2 の解答:

```

1 const spots = [
2     { name: '金閣寺', lat: 35.0394, lng: 135.7292 },
3     { name: '清水寺', lat: 34.9949, lng: 135.7850 },
4     { name: '伏見稻荷', lat: 34.9671, lng: 135.7727 }
5 ];
6
7 spots.forEach(spot => {
8     L.marker([spot.lat, spot.lng])
9         .addTo(map)
10        .bindPopup(`<b>${spot.name}</b>`);
11 });

```

問題 3 の解答:

```

1 spots.forEach((spot, index) => {
2     L.marker([spot.lat, spot.lng])
3         .addTo(map)

```

```
4     .bindPopup(`<b>${index + 1}. ${spot.name}</b>`);  
5 });
```


第6章

円を描画

6.1 学習内容

地図上に円（Circle）と円マーカー（CircleMarker）を描画する方法を学びます。

新しいメソッド・関数

`L.circle([lat, lng], options)` 半径をメートル単位で指定した円
`L.circleMarker([lat, lng], options)` 半径をピクセル単位で指定した円マーカー

6.2 コード解説

```

1 // 円を描画 (半径はメートル)
2 const circle = L.circle([35.6812, 139.7671], {
3   color: 'red',
4   fillColor: '#f03',
5   fillOpacity: 0.3,
6   radius: 500 // 500メートル
7 }).addTo(map);
8
9 // 円マーカーを描画 (半径はピクセル)
10 const circleMarker = L.circleMarker([35.6896, 139.7006], {
11   color: 'blue',
12   radius: 20 // 20ピクセル
13 }).addTo(map);

```

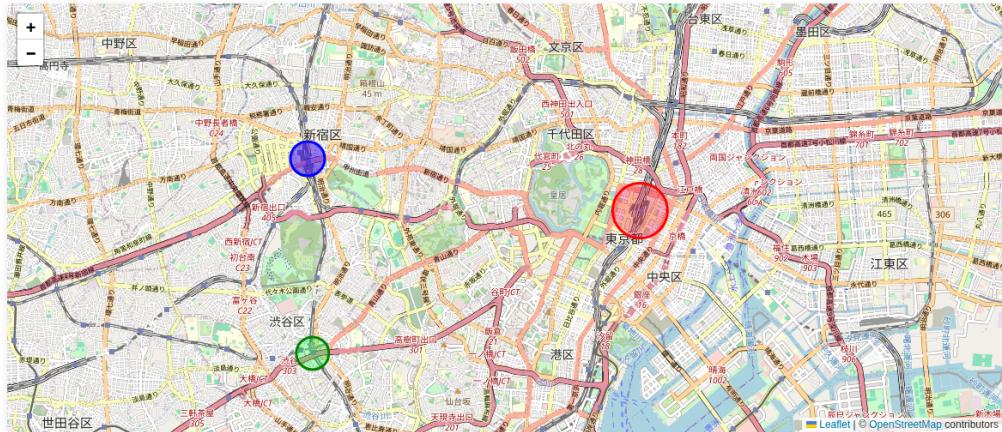
Listing 6.1 円の描画

ポイント

`L.circle()` はズームに応じてサイズが変化し、`L.circleMarker()` は常に同じピクセルサイズを維持します。

05 - 円を描画

学習内容: 地図上に円 (Circle) と円マーク (CircleMarker) を描画します。
ポイント:
 • L.circle() は半径をメートル単位で指定 (地図上で実際のサイズ)
 • L.circleMarker() は半径をピクセル単位で指定 (ズームに関係なく一定)
 • color, fillColor, fillOpacity などでスタイルを設定



```
// 円を描画 (半径はメートル)
const circle = L.circle([35.6812, 139.7671], {
  color: 'red',
  fillColor: '#f08'
});
```

図 6.1 サンプル実行結果

6.3 練習問題

練習問題

- 東京駅を中心に、半径 1000 メートルの緑色 (color: 'green') の円を描画してください。
- 上記の円の塗りつぶし透明度を 0.2 にしてください。
- 同じ位置に、半径 30 ピクセルの青い円マークを追加してください。
- 円と円マークそれぞれにポップアップを追加してください。

模範解答

問題 1~2 の解答:

```
1 L.circle([35.6812, 139.7671], {
2   color: 'green',
3   fillColor: 'green',
4   fillOpacity: 0.2,
5   radius: 1000
6 }).addTo(map);
```

問題 3 の解答:

```
1 L.circleMarker([35.6812, 139.7671], {
2   color: 'blue',
3   radius: 30
4 })
```

```
4 }).addTo(map);
```

問題 4 の解答:

```
1 L.circle([35.6812, 139.7671], {
2   color: 'green',
3   radius: 1000
4 }).addTo(map).bindPopup('半径1kmエリア');
5
6 L.circleMarker([35.6812, 139.7671], {
7   color: 'blue',
8   radius: 30
9 }).addTo(map).bindPopup('中心点');
```


第7章

ポリゴンを描画

7.1 学習内容

地図上に多角形（ポリゴン）と矩形（四角形）を描画する方法を学びます。

06 - ポリゴンを描画

学習内容: 地図上に多角形（ポリゴン）と矩形（四角形）を描画します。

ポイント:

- `L.polygon()` は座標の配列から多角形を作成
- `L.rectangle()` は対角の2点から矩形を作成
- 穴あきポリゴンも作成可能（配列のネスト）

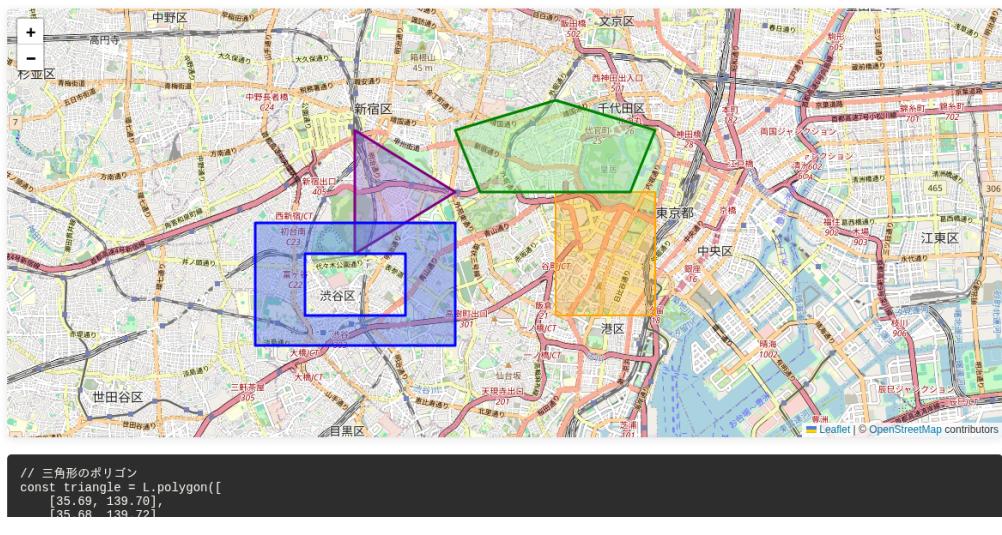


図 7.1 ポリゴンの描画

新しいメソッド・関数

`L.polygon(latlngs, options)` 座標の配列から多角形を作成
`L.rectangle(bounds, options)` 対角の2点から矩形を作成

7.2 コード解説

```
1 // 三角形のポリゴン
```

```

2 const triangle = L.polygon([
3     [35.69, 139.70],
4     [35.68, 139.72],
5     [35.67, 139.70]
6 ], {
7     color: 'purple',
8     fillOpacity: 0.4
9 }).addTo(map);
10
11 // 矩形
12 const rectangle = L.rectangle([
13     [35.66, 139.74], // 南西
14     [35.68, 139.76] // 北東
15 ], {
16     color: 'orange',
17     weight: 2
18 }).addTo(map);

```

Listing 7.1 ポリゴンの描画

7.3 練習問題

練習問題

1. 皇居周辺を囲む五角形のポリゴンを作成してください（任意の5座標を使用）。
2. 上記のポリゴンに紫色の枠線と、半透明の紫色の塗りつぶしを設定してください。
3. L.rectangle() を使って、新宿駅周辺に矩形を描画してください。

模範解答

問題1～2の解答:

```

1 L.polygon([
2     [35.690, 139.750],
3     [35.683, 139.745],
4     [35.680, 139.755],
5     [35.685, 139.765],
6     [35.692, 139.760]
7 ], {
8     color: 'purple',
9     fillColor: 'purple',
10    fillOpacity: 0.3
11 }).addTo(map);

```

問題3の解答:

```

1 L.rectangle([
2     [35.685, 139.695], // 南西
3     [35.695, 139.705] // 北東

```

```
4 ], {
5     color: 'blue',
6     fillOpacity: 0.2
7 }).addTo(map);
```


第8章

ポリラインを描画

8.1 学習内容

地図上に線（ポリライン）を描画してルートや経路を表現する方法を学びます。

07 - ポリラインを描画

学習内容: 地図上に線（ポリライン）を描画してルートや経路を表現します。

ポイント:

- `L.polyline()` で座標の配列から線を作成
- `weight` で線の太さ、`dashArray` で破線を設定
- 複数の線をマルチポリラインとして描画可能

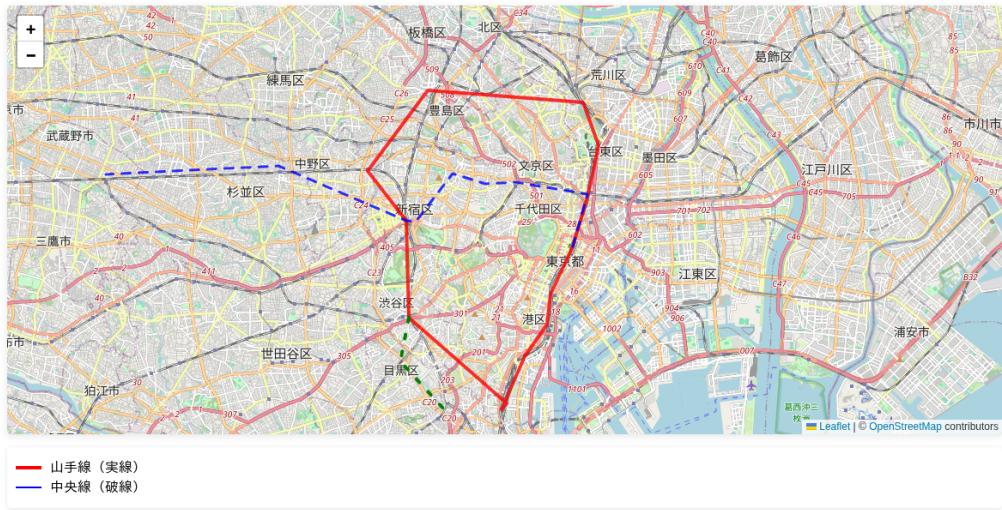


図 8.1 ポリラインの描画

新しいメソッド・関数

`L.polyline(latlngs, options)` 座標の配列から線を作成

8.1.1 主なスタイルオプション

`color` 線の色

`weight` 線の太さ（ピクセル）

`opacity` 不透明度 (0~1)
`dashArray` 破線パターン (例: '10, 10')

8.2 コード解説

```

1 // 山手線風のポリライン
2 const yamanoteLine = L.polyline([
3   [35.6812, 139.7671], // 東京
4   [35.7141, 139.7774], // 上野
5   [35.7321, 139.7090], // 池袋
6   [35.6896, 139.7006] // 新宿
7 ], {
8   color: 'red',
9   weight: 4,
10  opacity: 0.8
11}).addTo(map);
12
13 // 破線のポリライン
14 const chuoLine = L.polyline([...], {
15   color: 'blue',
16   dashArray: '10, 10' // 10px線、10px空白
17}).addTo(map);

```

Listing 8.1 ポリラインの描画

8.3 練習問題

練習問題

- 東京駅→上野駅→秋葉原駅→東京駅を結ぶ閉じたポリラインを作成してください。
- 上記のポリラインを太さ 5 ピクセル、オレンジ色で描画してください。
- 破線パターン (`dashArray: '15, 10, 5, 10'`) を使用したポリラインを作成してください。

模範解答

問題 1~2 の解答:

```

1 L.polyline([
2   [35.6812, 139.7671], // 東京駅
3   [35.7141, 139.7774], // 上野駅
4   [35.6983, 139.7731], // 秋葉原駅
5   [35.6812, 139.7671] // 東京駅（閉じる）
6 ], {
7   color: 'orange',
8   weight: 5

```

```
9 }).addTo(map);
```

問題 3 の解答:

```
1 L.polyline([
2     [35.6812, 139.7671],
3     [35.6896, 139.7006],
4     [35.6580, 139.7016]
5 ], {
6     color: 'purple',
7     weight: 3,
8     dashArray: '15, 10, 5, 10' // 長い線、間隔、短い線、間隔
9 }).addTo(map);
```


第9章

地図のズームとパン操作

9.1 学習内容

プログラムから地図のズームレベルや表示位置を制御する方法を学びます。

08 - 地図のズームとパン操作

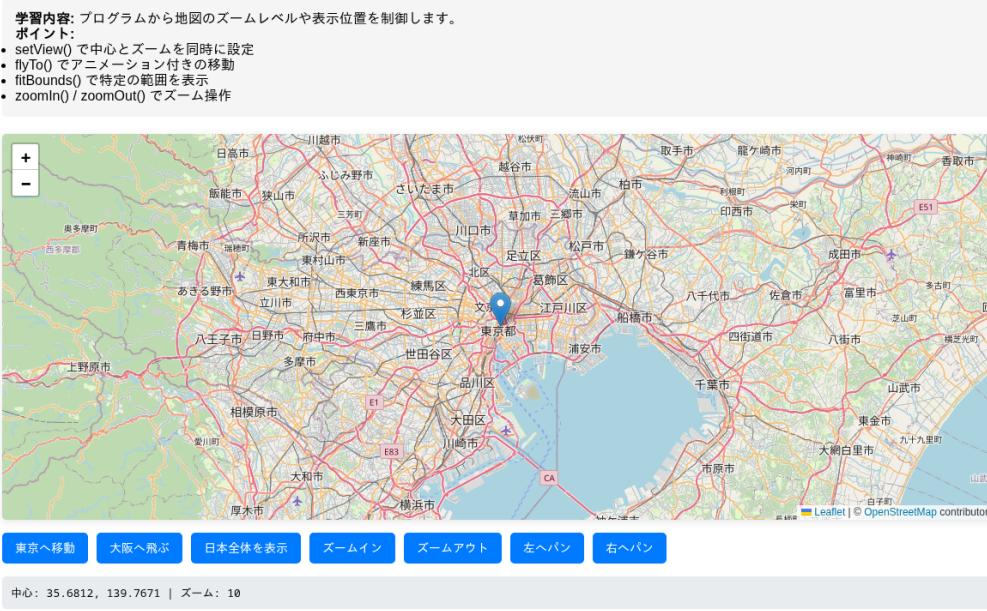


図 9.1 ズームとパン操作

新しいメソッド・関数

```
setView([lat, lng], zoom) 即座に中心とズームを設定
flyTo([lat, lng], zoom, options) アニメーション付きで移動
fitBounds(bounds) 指定範囲が収まるように表示
zoomIn() ズームイン
zoomOut() ズームアウト
panBy([x, y]) ピクセル単位で移動
getCenter() 現在の中心座標を取得
getZoom() 現在のズームレベルを取得
```

9.2 コード解説

```
1 // 即座に移動
2 map.setView([35.6812, 139.7671], 15);
3
4 // アニメーション付きで移動
5 map.flyTo([34.6937, 135.5023], 13, {
6   duration: 2 // 2秒かけてアニメーション
7 });
8
9 // 範囲にフィット
10 map.fitBounds([
11   [31.0, 129.0], // 南西
12   [45.5, 145.8] // 北東
13 ]);
```

Listing 9.1 地図の操作

9.3 練習問題

練習問題

- ボタンをクリックすると東京駅にアニメーション付きで移動する機能を実装してください。
- 地図の現在のズームレベルを取得し、コンソールに表示してください。
- `fitBounds()`を使って、東京駅と大阪駅の両方が収まるように表示してください。

模範解答

問題1の解答:

```
1 document.getElementById('tokyoBtn').addEventListener('click', () => {
2   map.flyTo([35.6812, 139.7671], 15, {
3     duration: 2
```

```
4  });
5 });
```

問題 2 の解答:

```
1 console.log('現在のズームレベル:', map.getZoom());
2 console.log('現在の中心座標:', map.getCenter());
```

問題 3 の解答:

```
1 map.fitBounds([
2   [34.7024, 135.4959], // 大阪駅（南西）
3   [35.6812, 139.7671] // 東京駅（北東）
4 ]);
```


第 10 章

異なるタイルレイヤー

10.1 学習内容

様々なタイルプロバイダーの地図を切り替えて表示する方法を学びます。

09 - 異なるタイルレイヤー

学習内容: 様々なタイルプロバイダーの地図を切り替えて表示します。

ポイント:

- OpenStreetMap以外にも多くのタイルプロバイダーがある
- 用途に応じて適切な地図スタイルを選択
- attribution（著作権表示）は必ず正しく設定

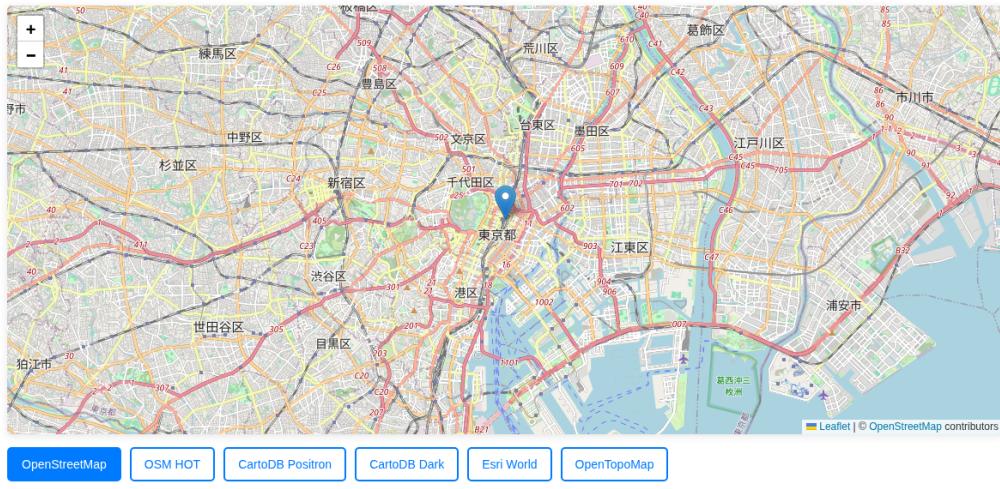


図 10.1 異なるタイルレイヤーの表示

ポイント

OpenStreetMap 以外にも多くのタイルプロバイダーがあります。用途に応じて適切な地図スタイルを選択しましょう。

10.2 主なタイルプロバイダー

OpenStreetMap 標準的な地図

CartoDB Positron 明るくシンプルなデザイン

CartoDB Dark Matter ダークテーマ

国土地理院 日本の詳細地図（標準地図・淡色地図）

10.3 コード解説

```

1 // CartoDB Positron
2 const cartoLight = L.tileLayer(
3     'https://s.basemaps.cartocdn.com/light_all/{z}/{x}/{y}.png',
4     { attribution: '...' }
5 );
6
7 // レイヤーの切り替え
8 map.removeLayer(currentLayer);
9 currentLayer = cartoLight;
10 currentLayer.addTo(map);

```

Listing 10.1 タイルレイヤーの切り替え

10.4 練習問題

練習問題

1. CartoDB Dark Matter タイルを使って、ダークテーマの地図を作成してください。
2. 国土地理院の標準地図タイルを使って、日本の詳細地図を表示してください。
3. ボタンクリックで異なるタイルレイヤーに切り替えられる機能を実装してください。

模範解答

問題 1 の解答:

```

1 L.tileLayer('https://s.basemaps.cartocdn.com/dark_all/{z}/{x}/{y}.png',
2             {
3                 attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors &copy; <a href="https://carto.com/attribution">CARTO</a>',
4             }).addTo(map);

```

問題 2 の解答:

```

1 L.tileLayer('https://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.png',
2             {
3                 attribution: '<a href="https://maps.gsi.go.jp/">国土地理院</a>',
4                 maxZoom: 18
5             }).addTo(map);

```

問題 3 の解答:

```
1 let currentLayer = osmLayer;
2 currentLayer.addTo(map);
3
4 document.getElementById('switchBtn').addEventListener('click', () => {
5     map.removeLayer(currentLayer);
6     currentLayer = (currentLayer === osmLayer) ? cartoLayer : osmLayer;
7     currentLayer.addTo(map);
8});
```


第 11 章

地図のイベント処理

11.1 学習内容

地図上でのクリック、ズーム、移動などのイベントを処理する方法を学びます。

10 - 地図のイベント処理

学習内容: 地図上でのクリック、ズーム、移動などのイベントを処理します。

ポイント:

- map.on() でイベントリスナーを登録
- クリック時に座標を取得
- ズームや移動の完了を検知
- e.latlng でクリック位置の座標を取得

操作: 地図をクリック、ズーム、移動してイベントログを確認してください。

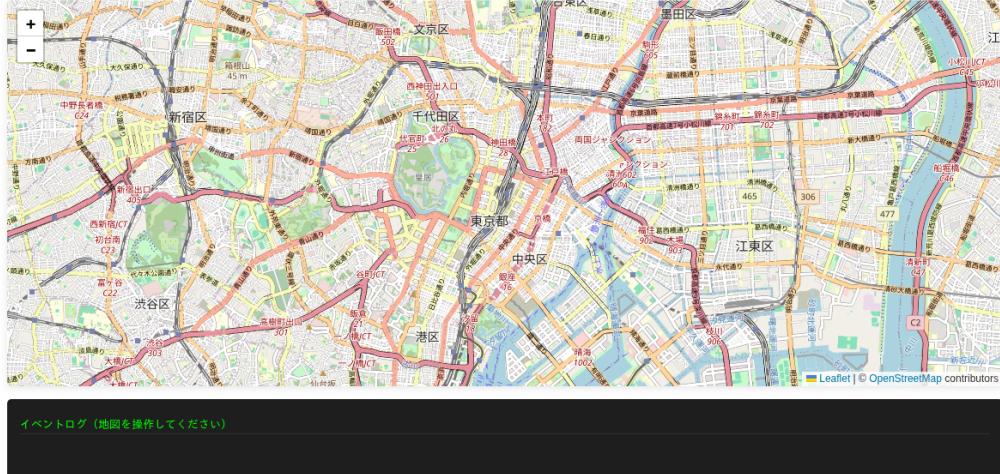


図 11.1 イベント処理の実装

新しいメソッド・関数

`map.on(event, callback)` イベントリスナーを登録

`map.off(event, callback)` イベントリスナーを解除

11.1.1 主なイベント

`click` クリック時

dblclick ダブルクリック時
 contextmenu 右クリック時
 zoomstart / zoomend ズーム開始/終了時
 movestart / moveend 移動開始/終了時
 mousemove マウス移動時

11.2 コード解説

```

1 // クリックイベント
2 map.on('click', function(e) {
3     // e.latlng でクリック位置の座標を取得
4     console.log('クリック位置:', e.latlng);
5     L.marker(e.latlng).addTo(map);
6 });
7
8 // ズームイベント
9 map.on('zoomend', function() {
10    console.log('ズームレベル:', map.getZoom());
11 });

```

Listing 11.1 イベント処理

11.3 練習問題

練習問題

- 地図をクリックするとその座標にマーカーを追加する機能を実装してください。
- 右クリック (contextmenu) でマーカーを追加し、通常クリックでは座標をポップアップで表示する機能を作成してください。
- 地図の移動終了時 (moveend) に、現在の中心座標をコンソールに表示する機能を追加してください。

模範解答

問題 1 の解答:

```

1 map.on('click', function(e) {
2     L.marker(e.latlng).addTo(map);
3 });

```

問題 2 の解答:

```

1 map.on('click', function(e) {
2     L.popup()
3         .setLatLng(e.latlng)
4         .setContent(`緯度: ${e.latlng.lat.toFixed(4)}, 経度: ${e.latlng.lng.toFixed(4)}`);
5 });

```

```
    .lng.toFixed(4})`)  
    .openOn(map);  
});  
  
8 map.on('contextmenu', function(e) {  
9     L.marker(e.latlng).addTo(map);  
10});
```

問題 3 の解答:

```
1 map.on('moveend', function() {  
2     const center = map.getCenter();  
3     console.log('中心座標:', center.lat, center.lng);  
4});
```


第 II 部

中級編

第 12 章

カスタムマーカーアイコン

12.1 学習内容

デフォルトのマーカーを独自のアイコンに変更する方法を学びます。

11 - カスタムマーカーアイコン

学習内容: デフォルトのマーカーを独自のアイコンに変更します。

ポイント:

- L.icon() で画像ベースのアイコンを作成
- L.divIcon() でHTMLベースのアイコンを作成
- iconSize, iconAnchor, popupAnchor でアイコンの配置を調整

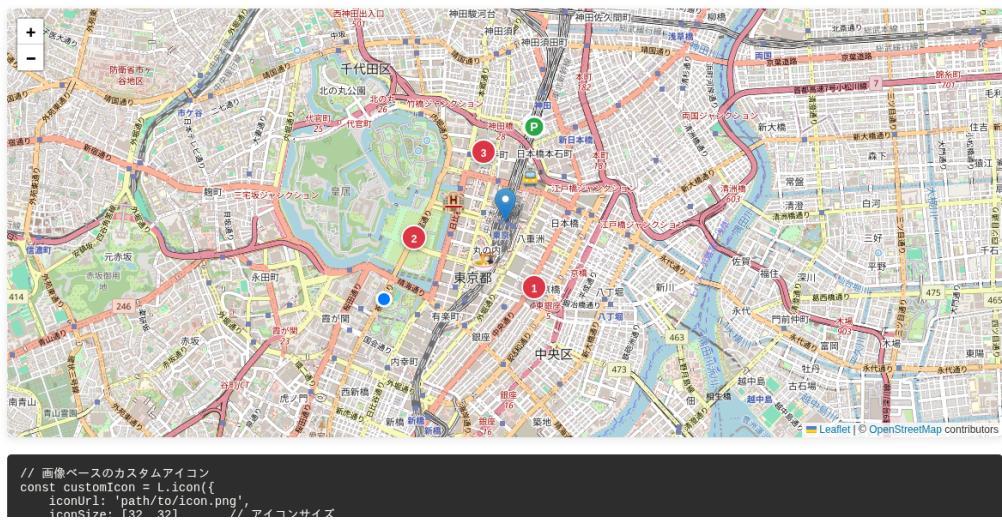


図 12.1 カスタムアイコンの表示

新しいメソッド・関数

L.icon(options) 画像ベースのアイコンを作成

L.divIcon(options) HTML ベースのアイコンを作成

12.1.1 主なオプション

iconUrl アイコン画像の URL

iconSize アイコンのサイズ [幅, 高さ]
 iconAnchor アイコンの基準点 [x, y]
 popupAnchor ポップアップの位置オフセット
 html (divIcon) 表示する HTML
 className (divIcon) CSS クラス名

12.2 コード解説

```

1 // HTMLベースのDivIcon（絵文字を使用）
2 const stationIcon = L.divIcon({
3   className: '',
4   html: '<div style="font-size: 24px;">✉</div>',
5   iconSize: [30, 30],
6   iconAnchor: [15, 15]
7 });
8
9 L.marker([35.68, 139.76], { icon: stationIcon }).addTo(map);

```

Listing 12.1 カスタムアイコン

12.3 練習問題

練習問題

1. 絵文字アイコン「✉」を使ったカスタムマーカーを作成してください。
2. 赤い丸（CSSで作成）をアイコンとして使用する DivIconを作成してください。
3. 複数の種類のアイコン（レストラン、ホテル、駅など）を使い分けてマーカーを配置してください。

模範解答

問題 1 の解答:

```

1 const homeIcon = L.divIcon({
2   className: '',
3   html: '<div style="font-size: 24px;">🏠</div>',
4   iconSize: [30, 30],
5   iconAnchor: [15, 15]
6 });
7 L.marker([35.68, 139.76], { icon: homeIcon }).addTo(map);

```

問題 2 の解答:

```

1 const redDotIcon = L.divIcon({
2   className: '',
3   html: '<div style="width: 16px; height: 16px; background: red; border-radius: 50%;"></div>';
4 });
5 L.marker([35.68, 139.76], { icon: redDotIcon }).addTo(map);

```

```
    border-radius: 50%; border: 2px solid white;"></div>',
4      iconSize: [20, 20],
5      iconAnchor: [10, 10]
6  });
7 L.marker([35.68, 139.76], { icon: redDotIcon }).addTo(map);
```

問題 3 の解答:

```
1 const icons = {
2   restaurant: L.divIcon({ html: '<div style="font-size:20px;">&#x1F374;</div>', iconSize: [24, 24], iconAnchor: [12, 12] }),
3   hotel: L.divIcon({ html: '<div style="font-size:20px;">&#x1F3E8;</div>', iconSize: [24, 24], iconAnchor: [12, 12] }),
4   station: L.divIcon({ html: '<div style="font-size:20px;">&#x1F689;</div>', iconSize: [24, 24], iconAnchor: [12, 12] })
5 };
6
7 L.marker([35.68, 139.76], { icon: icons.restaurant }).addTo(map);
8 L.marker([35.69, 139.70], { icon: icons.hotel }).addTo(map);
```


第 13 章

ドラッグ可能なマーカー

13.1 学習内容

ユーザーがドラッグして位置を変更できるマーカーを作成する方法を学びます。

12 - ドラッグ可能なマーカー



図 13.1 ドラッグ可能なマーカー

新しいメソッド・関数

```
draggable: true マーカーのドラッグを有効化
marker.getLatLng() 現在の座標を取得
marker.setLatLng([lat, lng]) 座標を設定
```

13.1.1 ドラッグイベント

dragstart ドラッグ開始時

drag ドラッグ中

dragend ドラッグ終了時

13.2 コード解説

```

1 const marker = L.marker([35.6812, 139.7671], {
2   draggable: true
3 }).addTo(map);
4
5 marker.on('dragend', function(e) {
6   const pos = marker.getLatLng();
7   console.log('最終位置:', pos.lat, pos.lng);
8 });

```

Listing 13.1 ドラッグ可能なマーカー

13.3 練習問題

練習問題

1. ドラッグ可能なマーカーを作成し、ドラッグ終了時にポップアップで座標を表示してください。
2. ドラッグ中 (drag イベント) にマーカーの現在位置をリアルタイムで画面に表示してください。
3. 「位置をリセット」ボタンをクリックすると、マーカーを初期位置に戻す機能を実装してください。

模範解答

問題 1 の解答:

```

1 const marker = L.marker([35.6812, 139.7671], { draggable: true }).addTo
  (map);
2 marker.on('dragend', function(e) {
3   const pos = marker.getLatLng();
4   marker.bindPopup(`緯度: ${pos.lat.toFixed(4)}<br>経度: ${pos.lng.
 toFixed(4)}`).openPopup();
5 });

```

問題 2 の解答:

```

1 const infoDiv = document.getElementById('position-info');

```

```
2 marker.on('drag', function(e) {  
3     const pos = marker.getLatLng();  
4     infoDiv.textContent = `緯度: ${pos.lat.toFixed(4)}, 経度: ${pos.lng  
    .toFixed(4)}`;  
5 });
```

問題 3 の解答:

```
1 const initialPos = [35.6812, 139.7671];  
2 const marker = L.marker(initialPos, {draggable: true}).addTo(map);  
3  
4 document.getElementById('resetBtn').addEventListener('click', () => {  
5     marker.setLatLng(initialPos);  
6 });
```


第 14 章

GeoJSON の基本

14.1 学習内容

GeoJSON 形式のデータを地図上に表示する方法を学びます。

13 - GeoJSONの基本

学習内容: GeoJSON形式のデータを地図上に表示します。

ポイント:

- GeoJSONは地理データの標準フォーマット
- Point, LineString, Polygonなどのジオメトリタイプ
- propertiesにメタデータを格納
- L.geoJSON()でGeoJSONを地図に追加

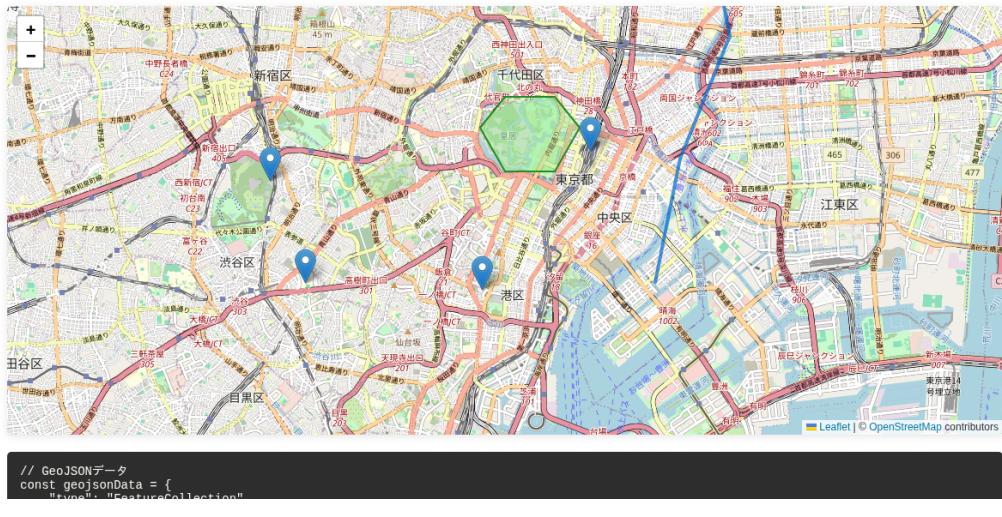


図 14.1 GeoJSON データの表示

ポイント

GeoJSON は地理データの標準フォーマットです。多くの GIS ツールや API が GeoJSON をサポートしています。

新しいメソッド・関数

`L.geoJSON(data, options)` GeoJSON レイヤーを作成

14.1.1 GeoJSON のジオメトリタイプ

Point 点
 LineString 線
 Polygon 多角形
 MultiPoint 複数点
 MultiLineString 複数線
 MultiPolygon 複数多角形

14.2 コード解説

```

1 const geojsonData = {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": { "name": "東京タワー" },
7       "geometry": {
8         "type": "Point",
9         "coordinates": [139.7454, 35.6586]
10      }
11    }
12  ]
13};
14
15 L.geoJSON(geojsonData).addTo(map);

```

Listing 14.1 GeoJSON の表示

注意

GeoJSON の座標は [経度, 緯度] の順番です。Leaflet の座標 [緯度, 経度] とは逆なので注意してください。

14.2.1 よくあるミス：座標の順序間違い

GeoJSON で座標を間違えると、地図が予想外の場所を表示します。

```

1 // 間違い：東京のつもりがアフリカ沖に表示される
2 "coordinates": [35.6586, 139.7454] // [緯度, 経度] は誤り
3
4 // 正しい：GeoJSONは [経度, 緯度] の順
5 "coordinates": [139.7454, 35.6586] // [経度, 緯度] が正解

```

Listing 14.2 座標順序のミス例

デバッグのコツ

地図がアフリカ大陸の西側（大西洋）を表示している場合、座標の順序が逆になっている可能性が高いです。日本の緯度（約 35 度）が経度として解釈されると、アフリカ付近になります。

14.3 練習問題

練習問題

- 3 つの Point フィーチャー（東京タワー、スカイツリー、東京駅）を含む GeoJSON を作成し、地図に表示してください。
- 各ポイントの `properties` に `name` を追加し、ポップアップで表示してください。
- LineString タイプの GeoJSON で東京の環状線を表現してください。

模範解答

問題 1～2 の解答:

```
1 const landmarks = {
2   "type": "FeatureCollection",
3   "features": [
4     { "type": "Feature", "properties": { "name": "東京タワー" },
5      "geometry": { "type": "Point", "coordinates": [139.7454,
6          35.6586] }},
7     { "type": "Feature", "properties": { "name": "スカイツリー" },
8      "geometry": { "type": "Point", "coordinates": [139.8107,
9          35.7101] }},
10    { "type": "Feature", "properties": { "name": "東京駅" },
11      "geometry": { "type": "Point", "coordinates": [139.7671,
12          35.6812] }}
13  ];
14
15 L.geoJSON(landmarks, {
16   onEachFeature: function(feature, layer) {
17     layer.bindPopup(feature.properties.name);
18   }
19 }).addTo(map);
```

問題 3 の解答:

```
1 const yamanoteLine = {
2   "type": "Feature",
3   "geometry": {
4     "type": "LineString",
5     "coordinates": [
6       [139.7671, 35.6812], [139.7774, 35.7141],
7       [139.7090, 35.7321], [139.7006, 35.6896]
```

```
8     ]
9   }
10  };
11 L.geoJSON(yamanoteLine, { style: { color: 'green', weight: 3 }}).addTo(
  map);
```

第 15 章

GeoJSON のスタイル

15.1 学習内容

GeoJSON データのプロパティに基づいて動的にスタイルを設定する方法を学びます。

14 - GeoJSON のスタイル

学習内容: GeoJSONデータのプロパティに基づいて動的にスタイルを設定します。
ポイント:
 • style関数でフィーチャーごとにスタイルを返す
 • propertiesの値に基づいて色や太さを変更
 • マウスオーバー時のインタラクティブなスタイル変更

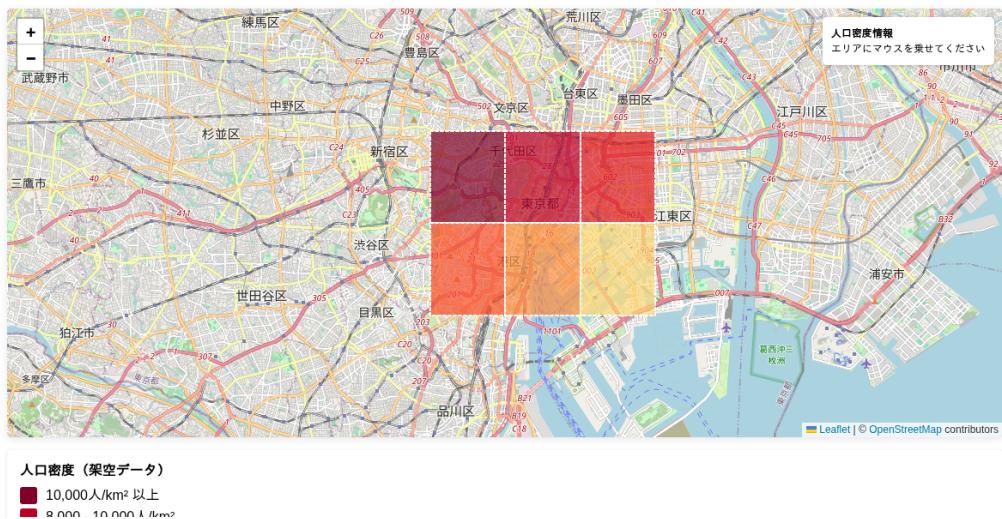


図 15.1 GeoJSON のスタイル

新しいメソッド・関数

style フィーチャーごとにスタイルを返す関数
onEachFeature 各フィーチャーに対する処理
pointToLayer ポイントの表示方法を定義
layer.setStyle(options) スタイルを変更
layer.resetStyle() スタイルをリセット
layer.bringToFront() 最前面に移動

15.2 コード解説

```

1 function getColor(density) {
2     return density > 10000 ? '#800026' :
3         density > 5000 ? '#E31A1C' :
4             '#FED976';
5 }
6
7 function style(feature) {
8     return {
9         fillColor: getColor(feature.properties.density),
10        weight: 2,
11        color: 'white',
12        fillOpacity: 0.7
13    };
14 }
15
16 L.geoJSON(data, { style: style }).addTo(map);

```

Listing 15.1 GeoJSON のスタイリング

15.3 練習問題

練習問題

- GeoJSON のプロパティ `type` ('restaurant', 'hotel', 'station') に基づいて、異なる色でスタイリングしてください。
- マウスオーバー時にポリゴンをハイライトする機能を実装してください。
- `onEachFeature` を使って、各フィーチャーにクリックイベントを追加してください。

模範解答

問題 1 の解答:

```

1 function style(feature) {
2     const colors = { restaurant: 'red', hotel: 'blue', station: 'green' };
3     return {
4         fillColor: colors[feature.properties.type] || 'gray',
5         weight: 2,
6         fillOpacity: 0.5
7     };
8 }
9 L.geoJSON(data, { style: style }).addTo(map);

```

問題 2 の解答:

```
1 function onEachFeature(feature, layer) {
2   layer.on({
3     mouseover: function(e) {
4       layer.setStyle({ weight: 5, color: '#666' });
5       layer.bringToFront();
6     },
7     mouseout: function(e) {
8       geojsonLayer.resetStyle(layer);
9     }
10   });
11 }
```

問題 3 の解答:

```
1 function onEachFeature(feature, layer) {
2   layer.on('click', function(e) {
3     alert('選択: ' + feature.properties.name);
4   });
5 }
6 L.geoJSON(data, { onEachFeature: onEachFeature }).addTo(map);
```


第 16 章

レイヤーグループ

16.1 学習内容

複数のレイヤーをグループ化して管理する方法を学びます。

15 - レイヤーグループ

学習内容: 複数のレイヤーをグループ化して管理します。

ポイント:

- `L.layerGroup()` で複数のレイヤーをまとめる
- `L.featureGroup()` は境界計算などの機能を追加
- グループ単位で表示/非表示を切り替え
- `addLayer()` / `removeLayer()` でレイヤーを追加/削除

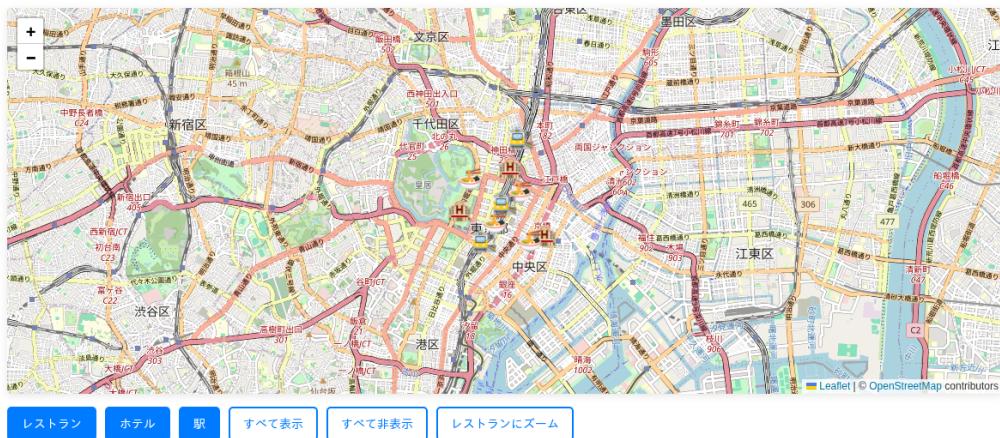


図 16.1 レイヤーグループの管理

新しいメソッド・関数

- `L.layerGroup()` レイヤーをグループ化
- `L.featureGroup()` 境界計算などの機能を追加
- `group.addLayer(layer)` レイヤーを追加
- `group.removeLayer(layer)` レイヤーを削除
- `map.hasLayer(layer)` レイヤーが存在するか確認
- `group.getBounds()` グループの境界を取得

16.2 コード解説

```

1 const restaurantsGroup = L.layerGroup();
2 const hotelsGroup = L.layerGroup();
3
4 // マーカーをグループに追加
5 L.marker([35.68, 139.76]).addTo(restaurantsGroup);
6
7 // グループを地図に追加
8 restaurantsGroup.addTo(map);
9
10 // グループの表示/非表示
11 if (map.hasLayer(restaurantsGroup)) {
12     map.removeLayer(restaurantsGroup);
13 } else {
14     map.addLayer(restaurantsGroup);
15 }

```

Listing 16.1 レイヤーグループ

16.3 練習問題

練習問題

- 「レストラン」「ホテル」「観光地」の 3 つのレイヤーグループを作成し、それぞれにマーカーを追加してください。
- ボタンクリックで特定のレイヤーグループの表示/非表示を切り替える機能を実装してください。
- L.featureGroup() を使って、すべてのマーカーを含む範囲に地図をフィットさせてください。

模範解答

問題 1 の解答:

```

1 const restaurantGroup = L.layerGroup();
2 const hotelGroup = L.layerGroup();
3 const spotGroup = L.layerGroup();
4
5 L.marker([35.68, 139.76]).addTo(restaurantGroup);
6 L.marker([35.69, 139.70]).addTo(hotelGroup);
7 L.marker([35.66, 139.75]).addTo(spotGroup);
8
9 [restaurantGroup, hotelGroup, spotGroup].forEach(g => g.addTo(map));

```

問題 2 の解答:

```
1 document.getElementById('toggleRestaurants').addEventListener('click',
2   () => {
3     if (map.hasLayer(restaurantGroup)) {
4       map.removeLayer(restaurantGroup);
5     } else {
6       map.addLayer(restaurantGroup);
7     }
7});
```

問題 3 の解答:

```
1 const allMarkers = L.featureGroup([...restaurantGroup.getLayers(),
2                                   ...hotelGroup.getLayers()]);
3 map.fitBounds(allMarkers.getBounds());
```


第 17 章

レイヤーコントロール

17.1 学習内容

Leaflet 標準のレイヤーコントロール UI を使用してベースマップとオーバーレイを切り替える方法を学びます。

16 - レイヤーコントロール

学習内容: Leaflet標準のレイヤーコントロールUIを使用してベースマップとオーバーレイを切り替えます。
ポイント:

- L.control.layers() でコントロールを作成。
- 第1引数: ベースレイヤー (ラジオボタン、排他的)
- 第2引数: オーバーレイ (チェックボックス、複数選択可)
- 地図右上のアイコンをクリックしてレイヤーを切り替え

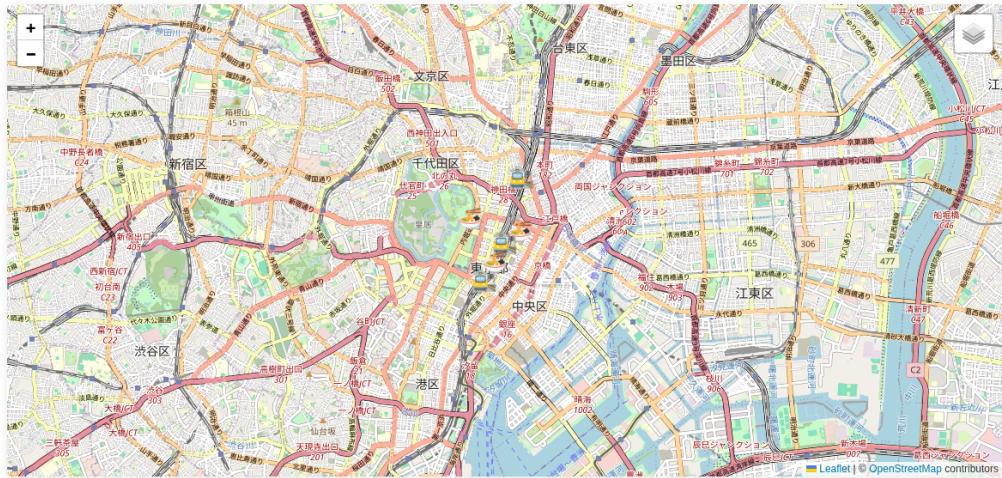


図 17.1 レイヤーコントロール

新しいメソッド・関数

```
L.control.layers(baseLayers, overlays, options) レイヤーコントロールを作成
```

17.1.1 主なオプション

collapsed true でマウスオーバー展開、false で常に展開

position 配置位置

17.2 コード解説

```

1 const baseLayers = {
2     "OpenStreetMap": osmLayer,
3     "CartoDB Light": cartoLight
4 };
5
6 const overlays = {
7     "レストラン": restaurantsGroup,
8     "ホテル": hotelsGroup
9 };
10
11 L.control.layers(baseLayers, overlays, {
12     collapsed: false,
13     position: 'topright'
14 }).addTo(map);

```

Listing 17.1 レイヤーコントロール

17.3 練習問題

練習問題

1. OpenStreetMap と CartoDB Dark Matter の 2 つのベースレイヤーを切り替えられるレイヤーコントロールを作成してください。
2. オーバーレイとして「マーカー」と「ポリゴン」を追加し、チェックボックスで表示/非表示を制御してください。
3. レイヤーコントロールを左下に配置し、常に展開した状態にしてください。

模範解答

問題 1~3 の解答:

```

1 const osm = L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}
2 .png', {
3     attribution: '&copy; <a href="https://www.openstreetmap.org/
4     copyright">OpenStreetMap</a> contributors'
5 }).addTo(map);
6
7 const cartoDark = L.tileLayer('https://s.basemaps.cartocdn.com/
8 dark_all/{z}/{x}/{y}.png', {
9     attribution: '&copy; <a href="https://www.openstreetmap.org/
10    copyright">OpenStreetMap</a> contributors &copy; <a href="https:
11 //carto.com/attribution">CARTO</a>'

```

```
7 });
8
9 const markersLayer = L.layerGroup([
10   L.marker([35.68, 139.76]),
11   L.marker([35.69, 139.70])
12 ]).addTo(map);
13
14 const polygonsLayer = L.layerGroup([
15   L.polygon([[35.68, 139.74], [35.69, 139.76], [35.67, 139.76]])
16 ]);
17
18 L.control.layers(
19   { "OpenStreetMap": osm, "Dark Mode": cartoDark },
20   { "マー カー": markersLayer, "ポリ ゴン": polygonsLayer },
21   { collapsed: false, position: 'bottomleft' }
22 ).addTo(map);
```


第 18 章

スケールコントロール

18.1 学習内容

地図にスケール（縮尺）やその他の標準コントロールを追加する方法を学びます。

17 - スケールコントロールとその他のコントロール

学習内容: 地図にスケール（縮尺）やその他の標準コントロールを追加します。

ポイント:

- L.control.scale() で縮尺バーを追加
- L.control.zoom() でズームボタン（デフォルトで表示）
- L.control.attribution() で著作権表示
- position オプションで配置を変更 (topright, topleft, bottomright, bottomleft)

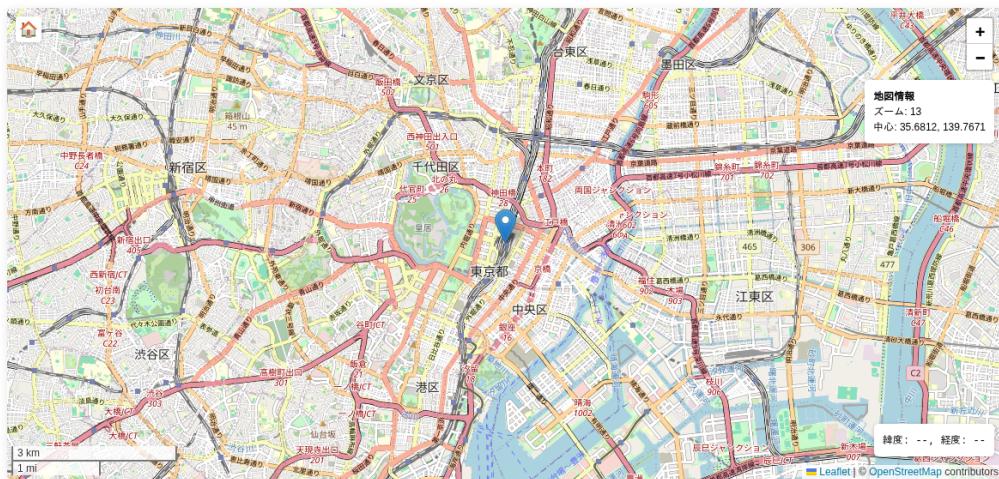


図 18.1 スケールコントロール

新しいメソッド・関数

L.control.scale(options) 縮尺バーを追加

L.control.zoom(options) ズームボタン

L.Control.extend(options) カスタムコントロールを作成

L.DomUtil.create(tag, className) DOM 要素を作成

L.DomEvent.disableClickPropagation(element) クリックの伝播を無効化

L.DomEvent.disableScrollPropagation(element) スクロールの伝播を無効化

イベント伝播の制御

カスタムコントロール内でフォーム入力やスクロール可能な要素を使う場合：

- `disableClickPropagation` : クリックが地図に伝播しない（必須）
- `disableScrollPropagation` : スクロールが地図のズームに影響しない

これらを設定しないと、コントロール内の操作が地図に影響してしまいます。

18.2 コード解説

```

1 L.control.scale({
2     maxWidth: 200,
3     metric: true,
4     imperial: true,
5     position: 'bottomleft'
6 }).addTo(map);
7
8 // カスタムコントロール
9 const CustomControl = L.Control.extend({
10     options: { position: 'bottomright' },
11     onAdd: function(map) {
12         const container = L.DomUtil.create('div', 'custom-control');
13         container.innerHTML = 'カスタム情報';
14         return container;
15     }
16 });
17 new CustomControl().addTo(map);

```

Listing 18.1 スケールコントロール

18.3 練習問題

練習問題

1. メートル表示のみのスケールコントロールを右下に追加してください。
2. 現在のズームレベルを表示するカスタムコントロールを作成してください（ズーム変更時に更新）。
3. 「現在地」ボタンを持つカスタムコントロールを作成し、クリックで地図の中心をリセットしてください。

模範解答

問題 1 の解答:

```

1 L.control.scale({

```

```
2     metric: true,
3     imperial: false,
4     position: 'bottomright'
5 }).addTo(map);
```

問題 2 の解答:

```
1 const ZoomInfo = L.Control.extend({
2   options: { position: 'topleft' },
3   onAdd: function(map) {
4     this._div = L.DomUtil.create('div', 'zoom-info');
5     this._div.style.cssText = 'padding: 5px 10px; background: white
6      ;';
7     this.update();
8     map.on('zoomend', () => this.update());
9     return this._div;
10  },
11  update: function() {
12    this._div.innerHTML = 'Zoom: ' + map.getZoom();
13  }
14});  
15 new ZoomInfo().addTo(map);
```

問題 3 の解答:

```
1 const HomeControl = L.Control.extend({
2   options: { position: 'topleft' },
3   onAdd: function(map) {
4     const btn = L.DomUtil.create('button', 'home-btn');
5     btn.innerHTML = '現在地';
6     L.DomEvent.disableClickPropagation(btn);
7     btn.onclick = () => map.setView([35.6812, 139.7671], 13);
8     return btn;
9   }
10});  
11 new HomeControl().addTo(map);
```


第 19 章

画像オーバーレイ

19.1 学習内容

地図上に画像や SVG をオーバーレイとして表示する方法を学びます。

18 - 画像オーバーレイ

学習内容: 地図上に画像やSVGをオーバーレイとして表示します。

ポイント:

- L.imageOverlay() で画像を地図上に配置
- bounds で表示範囲を指定
- opacity で透明度を調整
- L.svgOverlay() で SVG を表示



図 19.1 画像オーバーレイ

新しいメソッド・関数

```
L.imageOverlay(url, bounds, options) 画像オーバーレイを作成
L.svgOverlay(svg, bounds, options) SVG オーバーレイを作成
overlay.setOpacity(value) 透明度を設定
```

19.2 コード解説

```

1 const imageBounds = [
2     [35.6750, 139.7450], // 南西
3     [35.6900, 139.7650] // 北東
4 ];
5
6 const imageOverlay = L.imageOverlay(imageUrl, imageBounds, {
7     opacity: 0.7,
8     interactive: true
9 }).addTo(map);

```

Listing 19.1 画像オーバーレイ

19.3 練習問題

練習問題

- 東京駅周辺に古地図のイメージオーバーレイを配置してください（任意の画像 URL を使用）。
- スライダーでオーバーレイの透明度を 0~1 の間で変更できる機能を実装してください。
- 画像オーバーレイにクリックイベントを追加し、クリック時にポップアップを表示してください。

模範解答

問題 1 の解答:

```

1 const bounds = [[35.675, 139.760], [35.690, 139.775]];
2 const overlay = L.imageOverlay('old-map.png', bounds).addTo(map);
3 map.fitBounds(bounds);

```

問題 2 の解答:

```

1 const slider = document.getElementById('opacitySlider');
2 slider.addEventListener('input', function() {
3     overlay.setOpacity(this.value);
4 });

```

問題 3 の解答:

```

1 const overlay = L.imageOverlay('image.png', bounds, {
2     interactive: true
3 }).addTo(map);
4
5 overlay.on('click', function() {
6     L.popup()
7         .setLatLng(overlay.getBounds().getCenter())
8         .setContent('古地図オーバーレイ')
9         .openOn(map);

```

10 });

第 20 章

現在地表示 (Geolocation)

20.1 学習内容

ブラウザの位置情報 API を使用して現在地を地図上に表示する方法を学びます。

19 - モバイル対応と現在地表示

学習内容: ブラウザの位置情報APIを使用して現在地を地図上に表示します。

ポイント:

- map.locate() でブラウザの位置情報を取得
- locationfound イベントで位置取得成功時の処理
- locationerror イベントでエラー処理
- setView: true で自動的にその位置に移動

注意: ブラウザで位置情報の許可が必要です。

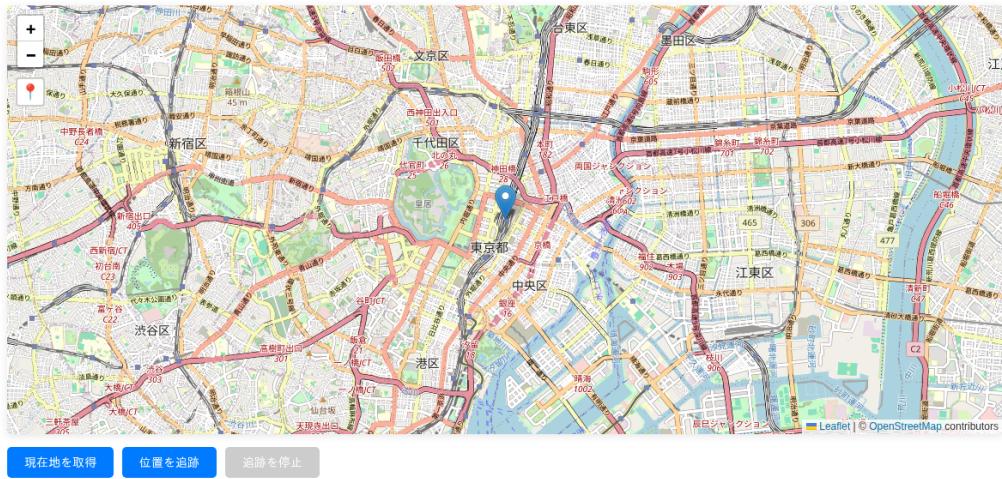


図 20.1 現在地の表示

新しいメソッド・関数

```
map.locate(options) 位置情報を取得  
map.stopLocate() 位置追跡を停止
```

20.1.1 主なオプション

setView true で取得位置に自動移動

maxZoom 自動移動時の最大ズーム
watch true で継続的に位置を追跡
enableHighAccuracy 高精度モード (GPS 使用)

20.1.2 位置情報イベント

locationfound 位置取得成功時
locationerror 位置取得失敗時

HTTPS 必須

ブラウザのセキュリティ仕様により、Geolocation API は **HTTPS 接続**または **localhost**でのみ動作します。**http://**や **file://**では位置情報の取得が拒否されます。

- 開発時 : localhost (VS Code Live Server、Python http.server 等) を使用
- 本番環境 : HTTPS 対応のサーバーにデプロイ

20.2 コード解説

```
1 map.locate({
2     setView: true,
3     maxZoom: 16,
4     enableHighAccuracy: true
5 });
6
7 map.on('locationfound', function(e) {
8     L.marker(e.latlng).addTo(map)
9         .bindPopup(`精度: ${e.accuracy}m`);
10    L.circle(e.latlng, { radius: e.accuracy }).addTo(map);
11 });
12
13 map.on('locationerror', function(e) {
14     alert('位置情報を取得できませんでした');
15 });
```

Listing 20.1 現在地表示

20.3 練習問題

練習問題

1. 「現在地を表示」ボタンをクリックすると、ユーザーの現在地にマーカーを表示する機能を実装してください。
2. 現在地の周囲に精度を示す円を描画してください。
3. 位置情報の取得に失敗した場合、エラーメッセージをポップアップで表示してください。

模範解答

問題 1~3 の解答:

```
1 let locationMarker = null;
2 let accuracyCircle = null;
3
4 document.getElementById('locateBtn').addEventListener('click', () => {
5     map.locate({ setView: true, maxZoom: 16 });
6 });
7
8 map.on('locationfound', function(e) {
9     if (locationMarker) {
10         locationMarker.remove();
11         accuracyCircle.remove();
12     }
13
14     locationMarker = L.marker(e.latlng).addTo(map)
15         .bindPopup(`現在地<br>精度: ${Math.round(e.accuracy)}m`)
16         .openPopup();
17
18     accuracyCircle = L.circle(e.latlng, {
19         radius: e.accuracy,
20         color: 'blue',
21         fillOpacity: 0.1
22     }).addTo(map);
23 });
24
25 map.on('locationerror', function(e) {
26     L.popup()
27         .setLatLng(map.getCenter())
28         .setContent('位置情報を取得できませんでした: ' + e.message)
29         .openOn(map);
30 });
```


第 21 章

ツールチップ

21.1 学習内容

マーカーやシェイプにホバー時のツールチップを追加する方法を学びます。

20 - ツールチップ

学習内容: マーカーやシェイプにホバー時のツールチップを追加します。

ポイント:

- bindTooltip() でツールチップをバインド
- permanent: true で常に表示
- direction で表示方向を指定 (top, bottom, left, right, center, auto)
- ポップアップとの違い: クリック不要でホバーで表示

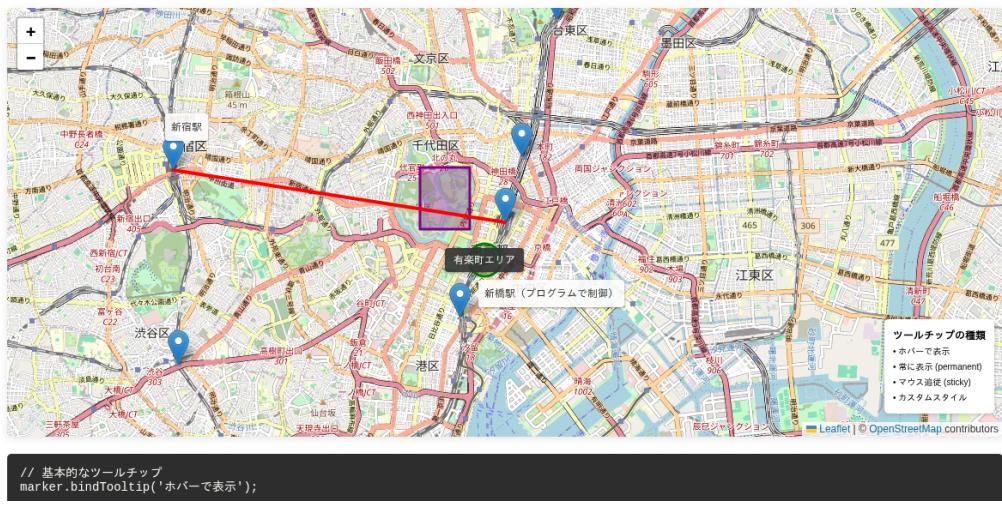


図 21.1 ツールチップの表示

新しいメソッド・関数

```
.bindTooltip(content, options) ツールチップをバインド
.openTooltip() ツールチップを開く
.closeTooltip() ツールチップを閉じる
```

21.1.1 主なオプション

`permanent true` で常に表示

`direction` 表示方向 (top, bottom, left, right, center, auto)

`sticky true` でマウスに追従

`offset` 位置オフセット [x, y]

`className` カスタム CSS クラス

21.2 コード解説

```

1 // 基本的なツールチップ
2 marker.bindTooltip('ホバーで表示');
3
4 // 常に表示
5 marker.bindTooltip('常に表示', {
6     permanent: true,
7     direction: 'top'
8 });
9
10 // マウスに追従（ポリゴン向け）
11 polygon.bindTooltip('エリア名', {
12     sticky: true
13 });

```

Listing 21.1 ツールチップ

21.3 練習問題

練習問題

1. マーカーに「東京駅」というツールチップを追加してください（ホバー時のみ表示）。
2. ツールチップを常に表示し、マーカーの上 (`direction: 'top'`) に配置してください。
3. ポリゴンにマウスに追従するツールチップを追加してください。

模範解答

問題 1 の解答:

```

1 L.marker([35.6812, 139.7671])
2     .addTo(map)
3     .bindTooltip('東京駅');

```

問題 2 の解答:

```

1 L.marker([35.6812, 139.7671])
2     .addTo(map)

```

```
3     .bindTooltip('東京駅', {  
4         permanent: true,  
5         direction: 'top',  
6         offset: [0, -10]  
7     });
```

問題 3 の解答:

```
1 L.polygon([  
2     [35.68, 139.74], [35.69, 139.76], [35.67, 139.76]  
3 ]).addTo(map).bindTooltip('このエリアをドラッグしてください', {  
4     sticky: true,  
5     direction: 'auto'  
6 });
```


第 III 部

上級編

第 22 章

コロプレスマップ

22.1 学習内容

データの値に応じて地域を色分けするコロプレスマップ（階級区分図）を作成する方法を学びます。

21 - コロプレスマップ（色分け地図）

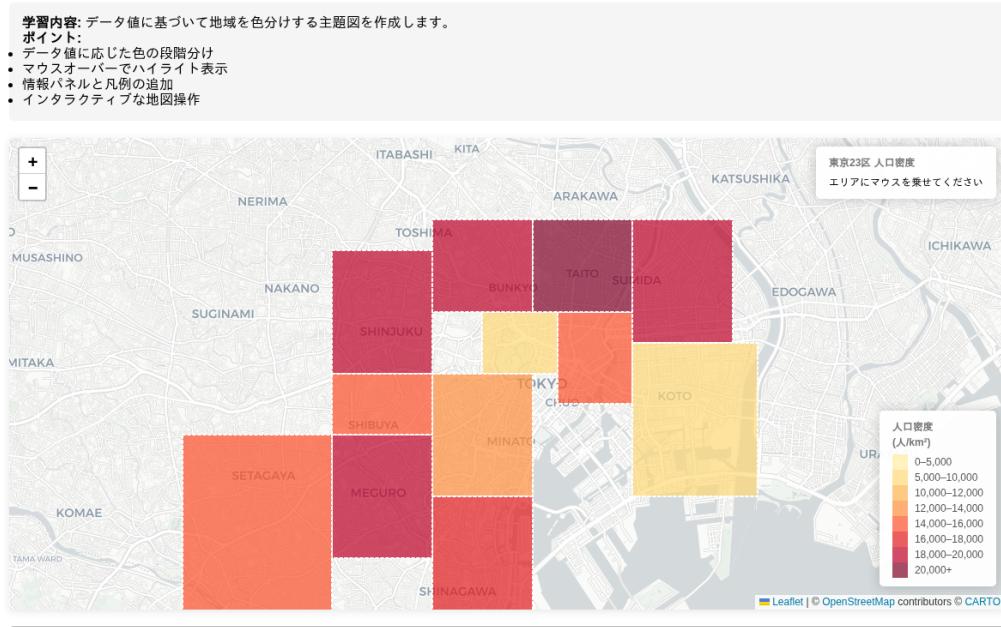


図 22.1 コロプレスマップの例

ポイント

コロプレスマップは、人口密度、所得水準、選挙結果などの統計データを視覚化するのに最適です。

22.2 主なテクニック

- 値に基づく色の段階分け (getColor 関数)

- マウスオーバー時のハイライト効果
- 情報パネルの更新
- 凡例の作成

22.3 練習問題

練習問題

- 人口密度に基づいて 5 段階の色分けを行う `getColor()` 関数を作成してください。
- マウスオーバー時に枠線を太くするハイライト効果を実装してください。
- 凡例コントロールを作成し、色と値の対応を表示してください。

模範解答

問題 1 の解答:

```

1 function getColor(density) {
2   return density > 10000 ? '#800026' :
3     density > 5000 ? '#BD0026' :
4     density > 2000 ? '#E31A1C' :
5     density > 1000 ? '#FC4E2A' :
6       '#FED976';
7 }
```

問題 2 の解答:

```

1 function highlightFeature(e) {
2   const layer = e.target;
3   layer.setStyle({ weight: 5, color: '#666', dashArray: '' });
4   layer.bringToFront();
5 }
6
7 function resetHighlight(e) {
8   geojsonLayer.resetStyle(e.target);
9 }
```

問題 3 の解答:

```

1 const legend = L.control({ position: 'bottomright' });
2 legend.onAdd = function() {
3   const div = L.DomUtil.create('div', 'legend');
4   const grades = [0, 1000, 2000, 5000, 10000];
5   div.innerHTML = '<strong>人口密度</strong><br>';
6   grades.forEach((grade, i) => {
7     div.innerHTML += `<i style="background:${getColor(grade + 1)}>
8       ${grade}${grades[i+1] ? '&ndash;' + grades[i+1] : '+'}</i>`;
9   });
10 }
```

```
9 |     return div;
10| };
11| legend.addTo(map);
```


第 23 章

カスタムコントロール

23.1 学習内容

`L.Control.extend()` を使って独自の UI コントロールを作成する方法を学びます。

22 - カスタムコントロールの作成

学習内容: `L.Control` を拡張して独自の UI コントロールを作成します。
 ポイント:
 • `L.Control.extend()` でコントロールクラスを作成
 • `onAdd()` で DOM 要素を返す
 • `onRemove()` でクリーンアップ処理
 • `L.DomEvent` でイベント処理

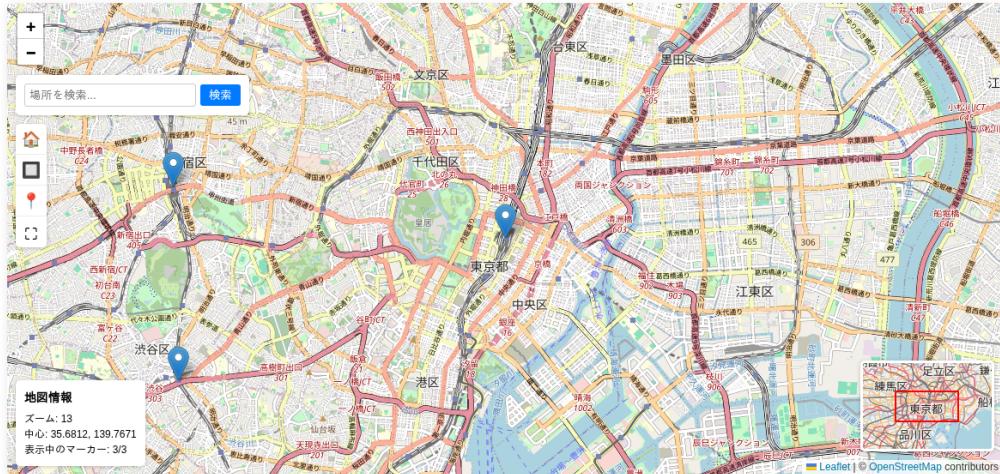


図 23.1 カスタムコントロール

新しいメソッド・関数

`L.Control.extend(options)` 既存コントロールを拡張
`onAdd(map)` コントロール追加時のコールバック
`onRemove(map)` コントロール削除時のコールバック

23.2 コード解説

```

1 const SearchControl = L.Control.extend({
2     options: { position: 'topleft' },
3     onAdd: function(map) {
4         const container = L.DomUtil.create('div', 'search-control');
5         const input = L.DomUtil.create('input', '', container);
6         input.placeholder = '検索...';
7
8         L.DomEvent.disableClickPropagation(container);
9
10    return container;
11 }
12 });
13
14 new SearchControl().addTo(map);

```

Listing 23.1 カスタムコントロール

23.3 練習問題

練習問題

1. 入力フィールドと検索ボタンを持つ検索コントロールを作成してください。
2. 地図の現在の中心座標とズームレベルを表示する情報コントロールを作成してください。
3. 複数のボタン（東京、大阪、名古屋）を持つナビゲーションコントロールを作成してください。

模範解答

問題 1 の解答:

```

1 const SearchControl = L.Control.extend({
2     options: { position: 'topleft' },
3     onAdd: function(map) {
4         const container = L.DomUtil.create('div', 'search-box');
5         container.innerHTML = '<input type="text" id="search"><button>
6             検索</button>';
7         L.DomEvent.disableClickPropagation(container);
8         return container;
9     }
10 });

```

問題 2 の解答:

```

1 const InfoControl = L.Control.extend({
2     onAdd: function(map) {
3         this._div = L.DomUtil.create('div', 'info');
4         map.on('moveend zoomend', () => this.update());

```

```
5     this.update();
6     return this._div;
7 },
8 update: function() {
9     const c = map.getCenter();
10    this._div.innerHTML = `中心: ${c.lat.toFixed(4)}, ${c.lng.
11       toFixed(4)}  
Zoom: ${map.getZoom()}`;
12 }
```


第 24 章

マーカークラスタリング（概念）

24.1 学習内容

多数のマーカーを効率的に表示するマーカークラスタリングの概念を学びます。

23 - マーカークラスタリングの概念

学習内容: 多数のマーカーをグループ化して表示するクラスタリングの基本概念を学びます。
ポイント:
 • 多数のマーカーを距離に基づいてグループ化
 • ズームレベルに応じてクラスタを分割/結合
 • クラスタアイコンで含まれるマーカー数を表示
 • 実際のプロジェクトでは Leaflet.markercluster プラグインを使用
注意: これは概念を理解するためのシンプルな実装です。本番では専用プラグインを推奨します。

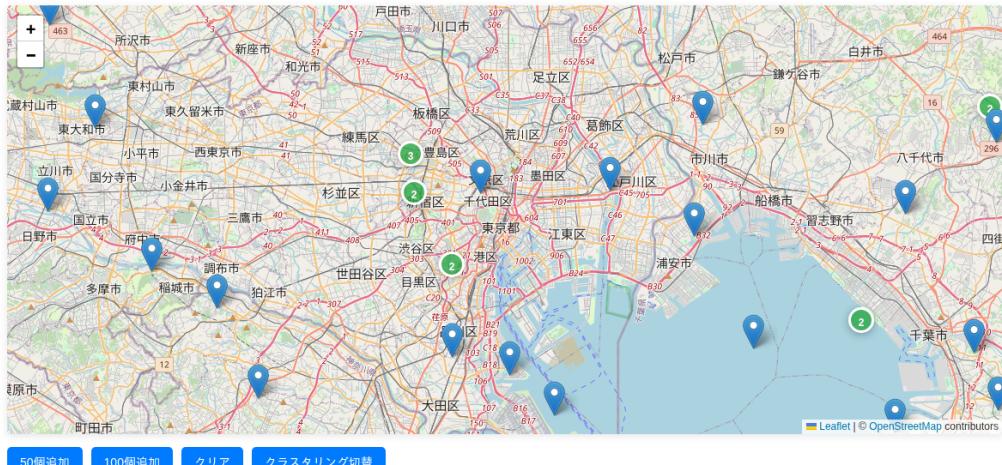


図 24.1 マーカークラスタリングの概念

ポイント

実際の実装には Leaflet.markercluster プラグインを使用することが一般的です。本章ではその概念と基本的なアルゴリズムを解説します。

24.2 クラスタリングの基本概念

- 近接するマーカーをグループ化

- ズームレベルに応じてクラスター/マーカーを切り替え
- クラスターアイコンにマーカー数を表示

24.3 練習問題

練習問題

- 100 個のランダムな位置にマーカーを配置してください。
- 距離に基づいて近いマーカーをグループ化する基本的なロジックを考えてください（概念のみ）。
- クラスターを表す DivIcon を作成し、マーカー数を表示してください。

模範解答

問題 1 の解答:

```

1 for (let i = 0; i < 100; i++) {
2   const lat = 35.6 + Math.random() * 0.2;
3   const lng = 139.6 + Math.random() * 0.3;
4   L.marker([lat, lng]).addTo(map);
5 }
```

問題 2 の解答:

```

1 // 概念的なグループ化ロジック
2 function groupNearbyMarkers(markers, threshold) {
3   const groups = [];
4   markers.forEach(marker => {
5     const nearGroup = groups.find(g =>
6       getDistance(g.center, marker.getLatLng()) < threshold
7     );
8     if (nearGroup) {
9       nearGroup.markers.push(marker);
10    } else {
11      groups.push({ center: marker.getLatLng(), markers: [marker] });
12    }
13  });
14  return groups;
15 }
```

問題 3 の解答:

```

1 function createClusterIcon(count) {
2   return L.divIcon({
3     html: `<div class="cluster-icon">${count}</div>`,
4     className: 'marker-cluster',
5     iconSize: [40, 40]
```

6 | });
7 | }

第 25 章

地図の境界制限

25.1 学習内容

地図の表示範囲やズームレベルを制限する方法を学びます。

24 - 地図の境界と制限

学習内容: 地図の表示範囲を制限し、ズームレベルを制御します。

学習内容
ポイント

- `maxBounds` で移動可能な範囲を制限
 - `minZoom / maxZoom` でズームレベルを制限
 - `getBounds()` で現在の表示範囲を取得
 - `fitBounds()` で指定範囲にフィット

操作: この地図は東京周辺に制限されています。範囲外への移動を試してみてください

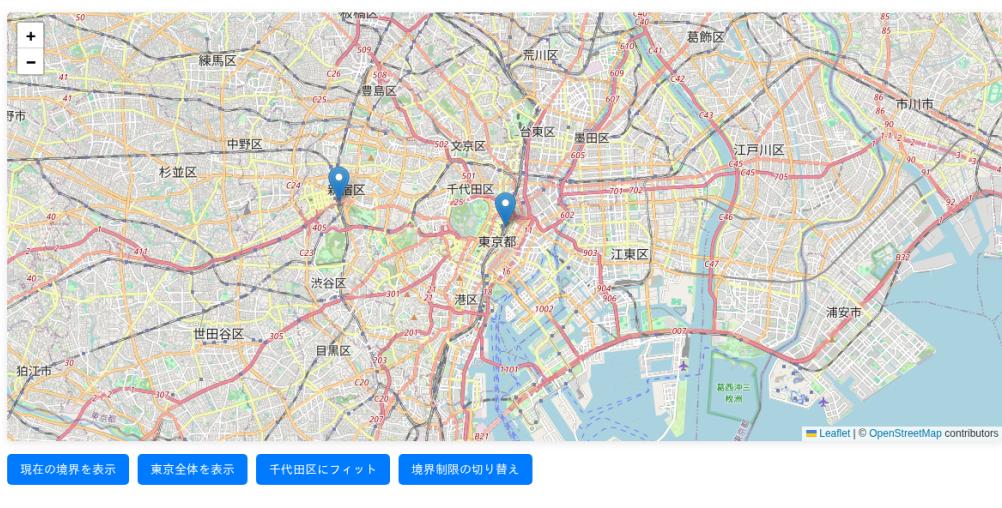


図 25.1 境界制限の設定

新しいメソッド・関数

L.latLngBounds(sw, ne) 境界オブジェクトを作成

`maxBounds` 表示可能な最大範囲

`maxBoundsViscosity` 境界の「粘り」(0~1)

`minZoom` / `maxZoom` ズームレベルの制限

`map.setMaxBounds(bounds)` 動的に境界を設定

25.2 コード解説

```

1 const tokyoBounds = L.latLngBounds(
2   [35.50, 139.50], // 南西
3   [35.85, 139.95] // 北東
4 );
5
6 const map = L.map('map', {
7   maxBounds: tokyoBounds,
8   maxBoundsViscosity: 1.0, // 完全に制限
9   minZoom: 10,
10  maxZoom: 18
11 });

```

Listing 25.1 境界制限

25.3 練習問題

練習問題

1. 東京 23 区の範囲に地図表示を制限してください。
2. ズームレベルを 10~16 に制限してください。
3. ボタンクリックで境界制限を解除・設定する機能を実装してください。

模範解答

問題 1~2 の解答:

```

1 const tokyo23Bounds = L.latLngBounds(
2   [35.53, 139.55], // 南西
3   [35.82, 139.92] // 北東
4 );
5
6 const map = L.map('map', {
7   maxBounds: tokyo23Bounds,
8   maxBoundsViscosity: 1.0,
9   minZoom: 10,
10  maxZoom: 16
11 }).setView([35.68, 139.76], 12);

```

問題 3 の解答:

```

1 let boundsEnabled = true;
2
3 document.getElementById('toggleBounds').addEventListener('click', () =>
4   {
    if (boundsEnabled) {

```

```
5     map.setMaxBounds(null);
6     map.setMinZoom(1);
7 } else {
8     map.setMaxBounds(tokyo23Bounds);
9     map.setMinZoom(10);
10 }
11 boundsEnabled = !boundsEnabled;
12});
```


第 26 章

動的なマーカー追加・削除

26.1 学習内容

ユーザー操作やデータに基づいてマーカーを動的に追加・削除・更新する方法を学びます。

25 - 動的なマーカー追加・削除

学習内容: ユーザー操作やデータに基づいてマーカーを動的に追加・削除・更新します。

- ポイント:
- `addTo()` / `remove()` でマーカーを追加・削除
 - `setLatLng()` で位置を更新
 - マーカーの参照を配列で管理
 - 地図クリックでマーカーを追加

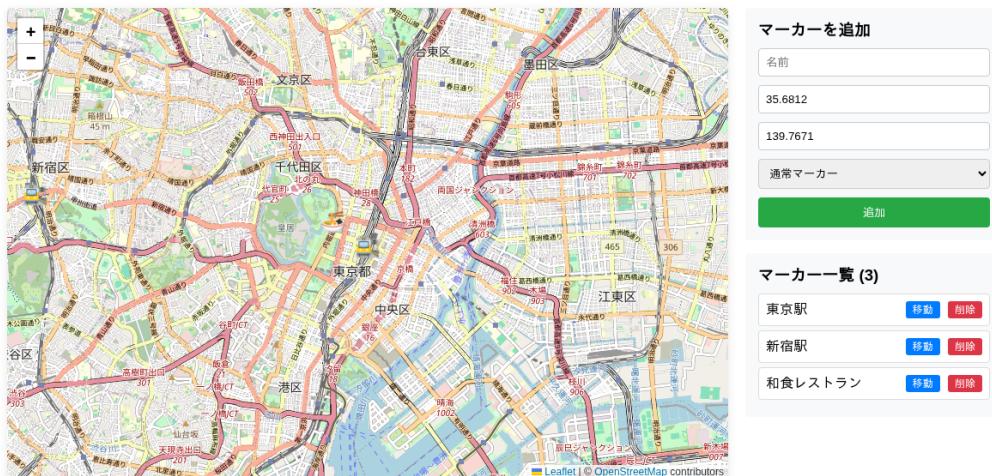


図 26.1 動的なマーカー管理

新しいメソッド・関数

```
marker.addTo(map) マーカーを追加
marker.remove() マーカーを削除
marker.setLatLng([lat, lng]) 位置を更新
marker.setIcon(icon) アイコンを変更
```

26.2 コード解説

```

1 let markers = [];
2
3 function addMarker(name, lat, lng) {
4     const marker = L.marker([lat, lng])
5         .addTo(map)
6         .bindPopup(name);
7     markers.push({ id: Date.now(), name, marker });
8 }
9
10 function removeMarker(id) {
11     const index = markers.findIndex(m => m.id === id);
12     if (index !== -1) {
13         markers[index].marker.remove();
14         markers.splice(index, 1);
15     }
16 }
```

Listing 26.1 動的なマーカー管理

26.3 練習問題

練習問題

1. 地図クリックで新しいマーカーを追加する機能を実装してください。
2. マーカーをクリックすると削除される機能を追加してください。
3. すべてのマーカーを削除する「クリア」ボタンを実装してください。

模範解答

問題 1 の解答:

```

1 map.on('click', function(e) {
2     const marker = L.marker(e.latlng).addTo(map);
3     markers.push(marker);
4 });
```

問題 2 の解答:

```

1 function addClickableMarker(latlng) {
2     const marker = L.marker(latlng).addTo(map);
3     marker.on('click', function() {
4         map.removeLayer(marker);
5         markers = markers.filter(m => m !== marker);
6     });
7     markers.push(marker);
```

```
8 }
```

問題 3 の解答:

```
1 document.getElementById('clearAll').addEventListener('click', () => {
2     markers.forEach(marker => map.removeLayer(marker));
3     markers = [];
4 });
```


第 27 章

アニメーション付きマーカー

27.1 学習内容

CSS アニメーションと JavaScript を使ってマーカーにアニメーション効果を追加する方法を学びます。

26 - アニメーション付きマーカー

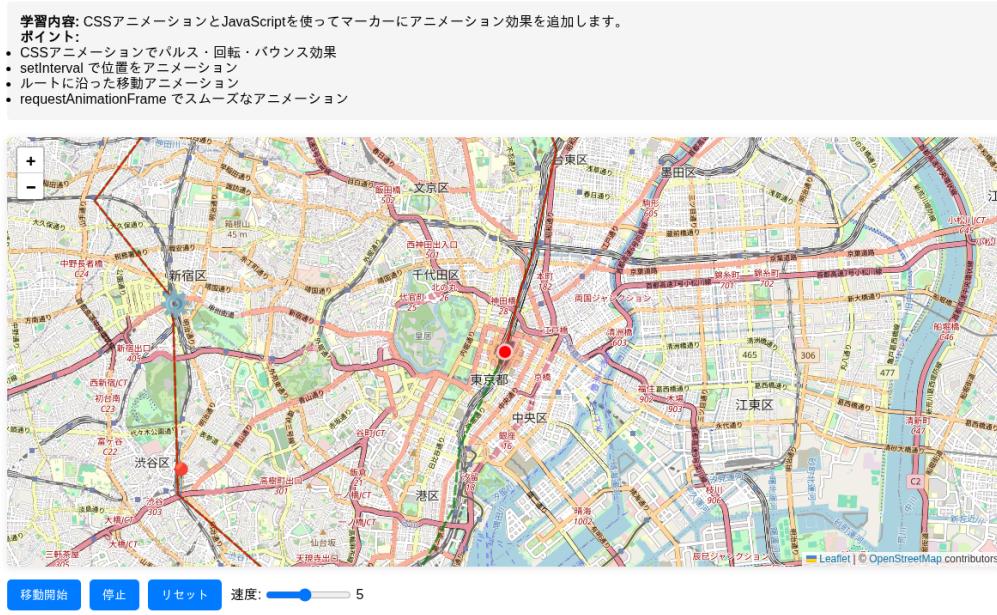


図 27.1 アニメーション付きマーカー

ポイント

- CSS アニメーションでパルス・回転・バウンス効果
- setInterval / setTimeout で位置をアニメーション
- ルートに沿った移動アニメーション

27.2 コード解説

```

1 const route = [[35.68, 139.76], [35.69, 139.77], ...];
2 let position = 0;
3
4 function animate() {
5     position += 1;
6     if (position >= route.length) position = 0;
7
8     marker.setLatLng(route[position]);
9     setTimeout(animate, 50);
10}
11
12 animate();

```

Listing 27.1 移動アニメーション

27.3 練習問題

練習問題

1. CSS アニメーションでパルス効果（拡大縮小）するマーカーを作成してください。
2. 2 点間を直線で移動するマーカーアニメーションを実装してください。
3. 複数のウェイポイントを順番に移動するルートアニメーションを作成してください。

模範解答

問題 1 の解答:

```

1 // CSS: .pulse { animation: pulse 1s infinite; }
2 // @keyframes pulse { 0%,100% { transform: scale(1); } 50% { transform:
3     scale(1.3); } }
4 const pulseIcon = L.divIcon({
5     html: '<div class="pulse" style="width:20px; height:20px;
6         background:red; border-radius:50%;"></div>',
7     iconSize: [20, 20]
8 });
9 L.marker([35.68, 139.76], { icon: pulseIcon }).addTo(map);

```

問題 2 の解答:

```

1 const start = [35.68, 139.76], end = [35.69, 139.77];
2 let progress = 0;
3 const marker = L.marker(start).addTo(map);
4
5 function animateMarker() {
6     progress += 0.01;

```

```
7  if (progress > 1) progress = 0;
8  const lat = start[0] + (end[0] - start[0]) * progress;
9  const lng = start[1] + (end[1] - start[1]) * progress;
10 marker.setLatLng([lat, lng]);
11 requestAnimationFrame/animateMarker();
12 }
13 animateMarker();
```


第 28 章

WMS レイヤー

28.1 学習内容

Web Map Service (WMS) を利用して地理空間データを地図上に表示する方法を学びます。

27 - WMS レイヤー

学習内容: Web Map Service (WMS) を利用して地理空間データを地図上に表示します。
ポイント:
 • L.tileLayer.wms() でWMSレイヤーを追加
 • layers パラメータで表示するレイヤーを指定
 • format で画像フォーマットを指定
 • transparent: true で背景を透過
注意: このサンプルはWMSの概念を説明するもので、実際のWMSサーバーへの接続例を示しています。

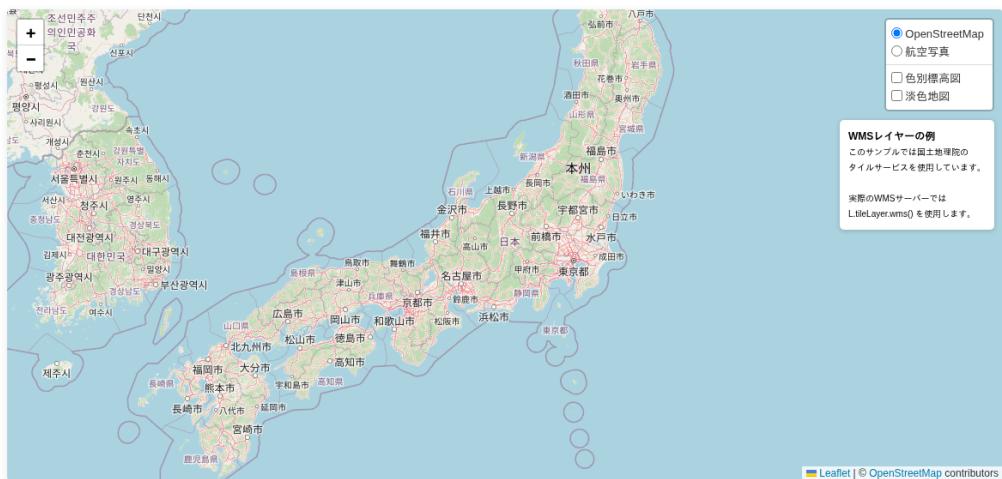


図 28.1 WMS レイヤーの表示

新しいメソッド・関数

```
L.tileLayer.wms(url, options) WMS レイヤーを作成
layer.setParams(params) パラメータを動的に変更
```

28.1.1 主なオプション

layers 表示するレイヤー名（カンマ区切りで複数可）

`format` 画像フォーマット (image/png 等)

`transparent` 背景透過

`version` WMS バージョン

`crs` 座標参照系 (デフォルト: 地図の CRS)

WMS 1.3.0 の座標軸順序に注意

WMS 1.3.0 では、EPSG:4326 (WGS84) の座標軸順序が緯度、経度 (lat, lon) になりました。これは WMS 1.1.1 以前の経度、緯度 (lon, lat) とは逆です。

Leaflet は内部でこの差異を吸収しますが、サーバーによっては正しく表示されないことがあります。その場合は以下を試してください：

- `version: '1.1.1'` に変更 (座標軸順序が lon, lat)
- サーバーの GetCapabilities を確認してサポートバージョンを特定

28.2 練習問題

練習問題

1. WMS レイヤーの基本的な構文を使って、気象データレイヤーを追加するコードを書いてください (概念的に)。
2. `transparent: true` オプションの効果を説明してください。
3. 複数の WMS レイヤーを切り替えられるレイヤーコントロールを作成してください。

模範解答

問題 1 の解答:

```

1 // 概念的な WMS レイヤー (実際のサーバー URL に置き換える)
2 const weatherLayer = L.tileLayer.wms('https://wms-server.example.com/
3   wms', {
4     layers: 'precipitation',
5     format: 'image/png',
6     transparent: true,
7     version: '1.3.0'
8 }).addTo(map);

```

問題 2 の解答: `transparent: true` を設定すると、WMS サーバーが返す画像の背景が透過されます。これにより、ベースマップの上にオーバーレイとして重ねて表示でき、下のレイヤーが見えるようになります。

問題 3 の解答:

```

1 const wmsLayers = {
2   "降水量": L.tileLayer.wms(wmsUrl, { layers: 'precipitation' }),
3   "気温": L.tileLayer.wms(wmsUrl, { layers: 'temperature' }),
4   "風速": L.tileLayer.wms(wmsUrl, { layers: 'wind' })

```

```
5 };  
6 L.control.layers(null, wmsLayers).addTo(map);
```


第 29 章

非地理的マップ

29.1 学習内容

Leaflet を使って地図以外のもの（フロアプラン、ゲームマップなど）を表示する方法を学びます。

28 - 非地理的マップ

学習内容: Leaflet を使って地図以外のもの（フロアプラン、ゲームマップなど）を表示します。
ポイント:
 • L.CRS.Simple を使用して単純な座標系を利用
 • 画像オーバーレイでカスタム画像を表示
 • ピクセル座標でマーカーやシェイプを配置
 • 用途: フロアプラン、ゲームマップ、座席表など

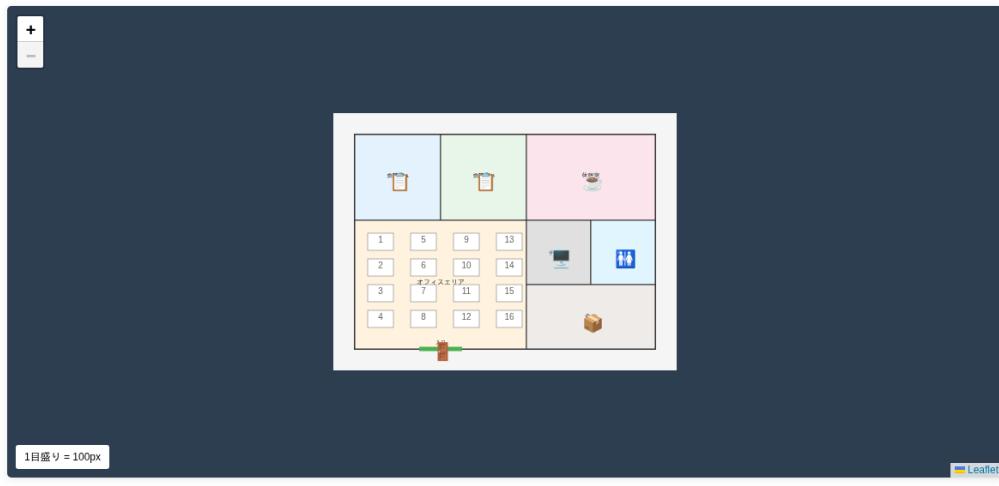


図 29.1 非地理的マップ

新しいメソッド・関数

L.CRS.Simple 単純な座標系（ピクセル座標）

29.2 コード解説

```
1 const map = L.map('map', {
2   crs: L.CRS.Simple,
```

```

3     minZoom: -1,
4     maxZoom: 2
5   );
6
7 const bounds = [[0, 0], [600, 800]];
8 L.imageOverlay('floorplan.png', bounds).addTo(map);
9 map.fitBounds(bounds);
10
11 // ピクセル座標でマークを配置
12 L.marker([300, 400]).addTo(map).bindPopup('会議室A');

```

Listing 29.1 非地理的マップ

注意

`L.CRS.Simple` では座標は `[y, x]` の順番です。また、SVG などとの座標系の違い（Y 軸の方向）に注意してください。

29.3 練習問題

練習問題

1. オフィスのフロアプランを表示する非地理的マップを作成してください（SVG または画像を使用）。
2. フロアプラン上の会議室の位置にマークを配置してください。
3. 複数フロアを切り替えるボタンを実装してください。

模範解答**問題 1 の解答:**

```

1 const map = L.map('map', {
2   crs: L.CRS.Simple,
3   minZoom: -2,
4   maxZoom: 2
5 });
6
7 const bounds = [[0, 0], [500, 700]];
8 L.imageOverlay('floor-plan.png', bounds).addTo(map);
9 map.fitBounds(bounds);

```

問題 2 の解答:

```

1 const rooms = [
2   { name: '会議室A', y: 100, x: 150 },
3   { name: '会議室B', y: 200, x: 350 },
4   { name: '休憩室', y: 400, x: 500 }
5 ];

```

```
6  
7 rooms.forEach(room => {  
8     L.marker([room.y, room.x]).addTo(map).bindPopup(room.name);  
9});
```

問題 3 の解答:

```
1 const floors = {  
2     '1F': L.imageOverlay('floor1.png', bounds),  
3     '2F': L.imageOverlay('floor2.png', bounds)  
4};  
5 let currentFloor = floors['1F'].addTo(map);  
6  
7 document.querySelectorAll('.floor-btn').forEach(btn => {  
8     btn.onclick = () => {  
9         map.removeLayer(currentFloor);  
10        currentFloor = floors[btn.dataset.floor].addTo(map);  
11    };  
12});
```


第30章

マップペイン

30.1 学習内容

マップペインを使ってレイヤーの重なり順を制御する方法を学びます。

29 - マップペイン

学習内容: マップペインを使ってレイヤーの重なり順を制御します。

ポイント:

- `map.createPane()` でカスタムペインを作成
- `z-index` でペインの重なり順を制御
- `pane` オプションでレイヤーを特定のペインに追加
- ラベルをポリゴンの上に表示するなど高度な制御が可能

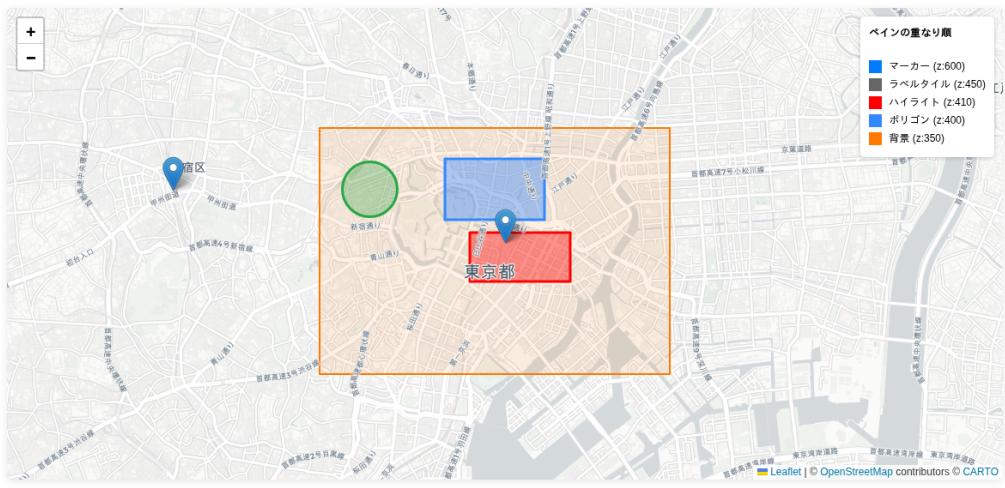


図 30.1 マップペインの制御

新しいメソッド・関数

```
map.createPane(name) カスタムペインを作成
map.getPane(name) ペインを取得
pane.style.zIndex 重なり順を設定
pane.style.pointerEvents クリックの透過設定
```

30.1.1 デフォルトのペイン構造

```
tilePane z-index: 200
overlayPane z-index: 400
shadowPane z-index: 500
markerPane z-index: 600
tooltipPane z-index: 650
popupPane z-index: 700
```

30.2 コード解説

```
1 // ラベル用ペインを作成 (z:450)
2 map.createPane('labelsPane');
3 map.getPane('labelsPane').style.zIndex = 450;
4 map.getPane('labelsPane').style.pointerEvents = 'none';
5
6 // ペインを使用してレイヤーを追加
7 L.tileLayer(labelTileUrl, {
8     pane: 'labelsPane'
9 }).addTo(map);
```

Listing 30.1 カスタムペイン

30.3 練習問題

練習問題

1. ポリゴンの上にラベルタイルが表示されるようにカスタムペインを作成してください。
2. ハイライト用ペイン (z-index: 410) を作成し、選択したポリゴンを強調表示してください。
3. pointerEvents: 'none' を設定して、ラベルをクリック透過させてください。

模範解答

問題 1 の解答:

```
1 map.createPane('labelsPane');
2 map.getPane('labelsPane').style.zIndex = 450;
3
4 L.tileLayer('https://s.basemaps.cartocdn.com/light_only_labels/{z}/{x
    }/{y}.png', {
5     pane: 'labelsPane'
6 }).addTo(map);
```

問題 2 の解答:

```
1 map.createPane('highlightPane');
2 map.getPane('highlightPane').style.zIndex = 410;
3
4 function highlightPolygon(polygon) {
5     L.polygon(polygon.getLatLngs(), {
6         pane: 'highlightPane',
7         color: 'yellow',
8         weight: 4,
9         fillOpacity: 0.3
10    }).addTo(map);
11}
```

問題 3 の解答:

```
1 map.createPane('labelsPane');
2 map.getPane('labelsPane').style.zIndex = 450;
3 map.getPane('labelsPane').style.pointerEvents = 'none';
4 // ラベルをクリックしても下のポリゴンが反応する
```


第 31 章

総合アプリケーション（店舗検索）

31.1 学習内容

これまで学んだ技術を組み合わせた実践的な店舗検索アプリケーションを作成します。

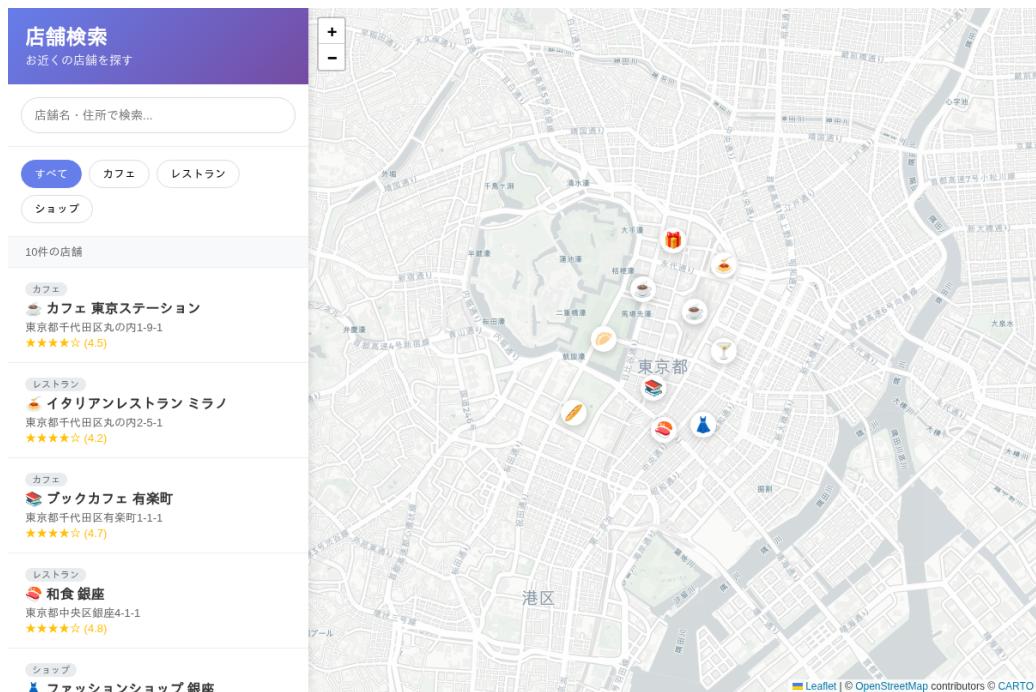


図 31.1 店舗検索アプリケーション実行結果

ポイント

使用技術（復習）：

- マーカーの動的追加・削除（第 26 章）
- カスタムアイコン（第 12 章）
- ポップアップ（第 4 章）
- イベント処理（第 11 章）
- レイヤーグループ（第 16 章）
- UI との連携

31.2 アプリケーション構成

1. 店舗データ：配列または API から取得
2. マーカー管理：オブジェクトで ID とマーカーを紐付け
3. フィルタリング：カテゴリと検索ワードで絞り込み
4. UI 連携：リストとマーカーを同期
5. インタラクション：クリックで選択、地図移動

31.3 コード解説

```

1 let markers = {};
2
3 function addMarkers() {
4     stores.forEach(store => {
5         if (currentCategory !== 'all' &&
6             store.category !== currentCategory) return;
7
8         const marker = L.marker([store.lat, store.lng], {
9             icon: createIcon(store.icon)
10        }).addTo(map);
11
12        marker.bindPopup(createPopupContent(store));
13        marker.on('click', () => selectStore(store.id));
14
15        markers[store.id] = marker;
16    });
17 }
18
19 function selectStore(storeId) {
20     const store = stores.find(s => s.id === storeId);
21     markers[storeId].setIcon(createIcon(store.icon, true));
22     markers[storeId].openPopup();
23     map.setView([store.lat, store.lng], 16);

```

24 }

Listing 31.1 店舗検索アプリの核心部分

31.4 練習問題

練習問題

1. 店舗データを配列で定義し、カテゴリ ('restaurant', 'cafe', 'shop') でフィルタリングする機能を実装してください。
2. 検索ボックスに店舗名を入力すると、該当する店舗のみが表示される機能を追加してください。
3. サイドバーの店舗リストとマーカーを連動させ、リストクリックで地図が移動する機能を実装してください。

模範解答

問題 1 の解答:

```
1 let currentCategory = 'all';
2
3 function filterByCategory(category) {
4     currentCategory = category;
5     clearMarkers();
6     stores.filter(s => category === 'all' || s.category === category)
7         .forEach(s => addMarker(s));
8 }
```

問題 2 の解答:

```
1 document.getElementById('searchInput').addEventListener('input', (e) =>
2 {
3     const query = e.target.value.toLowerCase();
4     clearMarkers();
5     stores.filter(s => s.name.toLowerCase().includes(query))
6         .forEach(s => addMarker(s));
7});
```

問題 3 の解答:

```
1 function renderStoreList() {
2     const listEl = document.getElementById('storeList');
3     listEl.innerHTML = stores.map(s => `
4         <div class="store-item" data-id="${s.id}">
5             ${s.name}
6         </div>
7     `).join('');
8
9     listEl.querySelectorAll('.store-item').forEach(el => {
```

```
10 el.onclick = () => {
11   const store = stores.find(s => s.id === el.dataset.id);
12   map.flyTo([store.lat, store.lng], 16);
13   markers[store.id].openPopup();
14 };
15 });
16 }
```

付録 A

よく使うオプション一覧

A.1 L.map() オプション

center 初期中心座標
 zoom 初期ズームレベル
 minZoom / maxZoom ズーム範囲
 maxBounds 表示可能範囲
 zoomControl ズームコントロールの表示
 scrollWheelZoom マウスホイールズーム
 doubleClickZoom ダブルクリックズーム
 dragging ドラッグ操作

A.2 L.tileLayer() オプション

attribution 著作権表示
 minZoom / maxZoom ズーム範囲
 opacity 透明度
 tileSize タイルサイズ
 subdomains サブドメインリスト

A.3 パスオプション（共通）

color 線の色
 weight 線の太さ
 opacity 線の不透明度
 fillColor 塗りつぶし色
 fillOpacity 塗りつぶし不透明度
 dashArray 破線パターン
 lineCap 線端の形状
 lineJoin 線接続の形状

付録 B

主要なタイルプロバイダー

B.1 タイル URL 一覧

OpenStreetMap

```
https://[s].tile.openstreetmap.org/{z}/{x}/{y}.png
```

CartoDB Positron

```
https://[s].basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png
```

CartoDB Dark Matter

```
https://[s].basemaps.cartocdn.com/dark_all/{z}/{x}/{y}{r}.png
```

国土地理院（標準地図）

```
https://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.png
```

国土地理院（淡色地図）

```
https://cyberjapandata.gsi.go.jp/xyz/pale/{z}/{x}/{y}.png
```

B.2 Attribution（著作権表示）テンプレート

タイルプロバイダーを使用する際は、必ず適切な attribution を設定してください。以下は各プロバイダーの推奨 attribution 例です。

```
1 // OpenStreetMap
2 attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">
3   OpenStreetMap</a> contributors'
4
5 // CartoDB
6 attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">
7   OpenStreetMap</a> contributors &copy; <a href="https://carto.com/
8     attributions">CARTO</a>'
```

Listing B.1 Attribution 設定例

利用時の注意

- **商用利用**：多くのプロバイダーは非商用・少量アクセスは無料ですが、商用利用や大量アクセスには制限がある場合があります。必ず各プロバイダーの利用規約を確認してください。
- **直接アクセスの回避**：大規模サービスでは、タイルをキャッシュするか、商用タイルサービス（Mapbox、MapTiler 等）の利用を検討してください。
- **リンク付き attribution**：attribution にはプロバイダーへのリンクを含めることが多くの規約で求められています。

付録 C

参考文献・参考サイト

C.1 公式ドキュメント

Leaflet.js 公式サイト

<https://leafletjs.com/>

Leaflet.js の公式ドキュメント。API リファレンス、チュートリアル、プラグイン一覧などが掲載されている。

Leaflet.js GitHub リポジトリ

<https://github.com/Leaflet/Leaflet>

ソースコード、イシュートラッカー、コントリビューションガイドラインなど。

Leaflet.js API Reference

<https://leafletjs.com/reference.html>

すべてのクラス、メソッド、オプションの詳細なリファレンス。

C.2 地図データ・タイルプロバイダー

OpenStreetMap

<https://www.openstreetmap.org/>

オープンソースの地図プロジェクト。本ガイドブックで使用しているタイルレイヤーの主要なソース。

国土地理院 地理院タイル

<https://maps.gsi.go.jp/development/ichiran.html>

日本の国土地理院が提供する各種地図タイル。標準地図、淡色地図、航空写真などが利用可能。

CartoDB Basemaps

<https://carto.com/basemaps/>

Positron (ライト)、Dark Matter (ダーク) などのスタイリッシュなベースマップを提供。

Stamen Design

<http://maps.stamen.com/>

Toner、Terrain、Watercolor などのアーティスティックな地図タイルを提供。

C.3 GeoJSON・地理データ

GeoJSON 仕様 (RFC 7946)

<https://tools.ietf.org/html/rfc7946>

GeoJSON フォーマットの公式仕様書。

geojson.io

<https://geojson.io/>

GeoJSON データをブラウザ上で作成・編集・可視化できる Web ツール。

Natural Earth

<https://www.naturalearthdata.com/>

世界の国境、都市、海岸線などのパブリックドメイン地理データセット。

C.4 Leaflet プラグイン

Leaflet.markercluster

<https://github.com/Leaflet/Leaflet.markercluster>

大量のマーカーをクラスタリングして表示するプラグイン。

Leaflet.draw

<https://github.com/Leaflet/Leaflet.draw>

地図上で図形を描画・編集するためのプラグイン。

Leaflet.fullscreen

<https://github.com/Leaflet/Leaflet.fullscreen>

フルスクリーン表示機能を追加するプラグイン。

Leaflet Plugins 一覧

<https://leafletjs.com/plugins.html>

Leaflet.js 公式サイトで紹介されているプラグインの一覧。

C.5 Web 技術・標準仕様

MDN Web Docs - Geolocation API

https://developer.mozilla.org/ja/docs/Web/API/Geolocation_API

ブラウザの Geolocation API についての詳細なドキュメント。

MDN Web Docs - JavaScript

<https://developer.mozilla.org/ja/docs/Web/JavaScript>

JavaScript 言語の包括的なリファレンス。

Web Map Service (WMS) 仕様

<https://www.ogc.org/standards/wms>

OGC (Open Geospatial Consortium) による WMS 標準仕様。

C.6 書籍・文献

『Leaflet.js Essentials』

著者: Paul Crickard III

出版: Packt Publishing, 2014

Leaflet.js の基礎から応用までを解説した入門書。

『Web Mapping Illustrated』

著者: Tyler Mitchell

出版: O'Reilly Media, 2005

Web 地図技術の基礎概念を解説した古典的な書籍。

『GeoJSON データでつくる地図』

各種技術ブログや Qiita 記事

日本語で GeoJSON と Leaflet.js の使い方を解説した記事が多数公開されている。

C.7 チュートリアル・学習リソース

Leaflet.js 公式チュートリアル

<https://leafletjs.com/examples.html>

公式サイトで提供されているステップバイステップのチュートリアル集。

MapSchool

<https://mapschool.io/>

地図と GIS の基礎概念を学べる無料のオンラインリソース。

Leaflet Provider Demo

<https://leaflet-extras.github.io/leaflet-providers/preview/>

様々なスタイルプロバイダーをプレビューできるデモサイト。

付録 D

LLM で学ぶプロンプトセット（20 本）

この章では、LLM（ChatGPT、Claude 等）を使って Leaflet.js を効率的に学ぶためのプロンプト集を紹介します。各プロンプトをコピペして使うことで、動作するコードと解説を得られます。

LLM 利用時の注意

LLM の出力は便利ですが、誤りを含むことがあります。必ずブラウザのコンソールや動作で確認してください。また、API キー、パスワード、個人情報、学内限定情報などはプロンプトに貼り付けないでください。不具合相談（P20）では、必要最小限のコードとログだけを共有し、機密情報は伏せてください。タイルや外部 API の利用規約・レート制限にも注意してください（付録 B 参照）。

D.1 難易度の目安

難易度	内容
★ 1	地図の表示・基本操作（最初の壁を超える）
★ 2	イベント（クリック等）・線/面・レイヤ切替
★ 3	状態管理（選択）・外部 UI 連携・データ取得の入口
★ 4	テーマ地図・大量点・プラグイン・地図 × 表
★ 5	範囲内取得・差分更新・性能比較

表 D.1 難易度の目安

D.2 LLM の使い方（失敗しにくい手順）

- まず P01 を投げて、出てきた HTML をそのままコピペして動かす
- 動いたら、次に P02 → P03…と順番に進む（時間が限られる場合は本章末尾の「おすすめの進め方」を参照）
- 途中で詰まつたら P20（デバッグ）に「症状」「コンソールログ」「コード」を貼る
- fetch を使う回（P11/P12/P17）は、file:///直開きだと動かないことが多いので、VS

Code の Live Server 等で `http://localhost` で開く（重要。詳しくは 1.4.1 節を参照）

5. 「動いた=OK」ではなく、**DevTools (Console/Network)** で確認する習慣をつける（特に `fetch` 回）

D.3 品質ブースター（任意）

以下のテキストを各プロンプトの前に付けると、LLM の出力が安定します。

LLM プロンプト

あなたは Leaflet.js の講師です。初学者が迷わないように、説明多めで作ってください。

条件:

- まず「ブラウザでそのまま動く 1 ファイル（HTML1 枚）」を必ず提示（CDN 利用 OK）
- コードは「コピペで動く」ことを最優先（不足が出ないように）
- その後に「なぜそう書くか」を短く説明
- つまづきやすい点（例: CSS 高さ、CORS、初期化順）を必ず注意書き

出力:

- 1) 到達目標
- 2) 完全な HTML（1 ファイル）
- 3) 動作確認チェックリスト
- 4) よくある失敗と直し方
- 5) 発展（次に何を足せるか）

品質ブースターの例外

原則として「HTML1 枚で動くコード」を出させるための前置きですが、P19 (Vite + TypeScript) のように環境構築や複数ファイルが前提の回では、プロンプト側の要件を優先してください（最小構成のフォルダ一式：`main.ts / index.html / package.json` など）。

D.4 P01（★1）最小構成で地図を表示

到達目標

- Leaflet で地図が表示できる（最初の成功体験）
- 「地図が出ない」最大原因の CSS 高さ問題を回避できる
- タイルと attribution（著作権表示）の意味が分かる

何を作る？ OSM 地図が表示される、最小の HTML1 枚。

注意

`#map` の高さが 0 だと地図は表示されません（最頻出）。JS より CSS が原因のことが多いです。

プレースホルダについて

プロンプト内の <lat> や <lng> は、あなたが置き換える値（テンプレート用）です。山括弧 <> で囲まれた部分を自分の値に置換してください。Leaflet のタイル URL に登場する {z}/{x}/{y} は波括弧 {} で Leaflet 側の予約変数なので、そのまま残してください。

LLM プロンプト

Leaflet で地図を表示する最小 HTML (1 ファイル) を作って。

要件:

- OSM タイル
- 初期中心: <lat>,<lng> (例: 34.7100, 137.7260) ← あなたの座標に置換
- ズーム: <z> (例: 13) ← あなたのズームレベルに置換
- CSS で地図コンテナの高さ問題を確実に回避 (html/body/#map の高さ指定を含める)
- attribution (著作権表示) は消さず、追記だけする
- 「どこを変更すれば中心やズームを変えられるか」をコメントで示す

D.5 P02 (★ 1) 配列からマーカー生成+ポップアップ+ツールチップ

到達目標

- 配列データをループしてマーカーを作れる
- tooltip (ホバー) と popup (クリック) の違いが分かる
- fitBounds で「全体が見える初期表示」ができる

何を作る？ 3~5 件の地点データを配列で持ち、まとめて表示。

注意

fitBounds は「座標の配列」や「LayerGroup」から範囲を作る必要あり。

LLM プロンプト

Leaflet で「配列データからマーカーを生成」し、tooltip と popup を付ける例を作って。

要件:

- データ配列: [{name, desc, lat, lng}, ...] をコード内に用意 (日本語 OK)
- tooltip はホバーで表示、popup はクリックで表示
- 全マーカーが画面内に入るよう fitBounds
- クリックしたマーカーのデータをコンソールにも出す
- 初学者向けに、tooltip/popup/fitBounds の役割を短く説明

D.6 P03 (★ 2) クリックで座標取得 & コピー

到達目標

- 地図クリックイベントから緯度経度を取得できる
- 取得した値を UI に表示できる
- クリップボードコピーの基本（成功/失敗）を扱える

何を作る？ クリック地点の座標を表示してコピーする「座標取得ツール」。

注意

クリップボード API は https://localhost 以外だと制限されることがある。

LLM プロンプト

地図クリックで緯度経度を取得し、(1) ポップアップ表示 (2) クリップボードコピー を実装して。

要件:

- 小数点 6 桁
- 右上に「最後にクリックした座標」を出す小パネル
- 「コピーしました」トースト表示（簡易で OK）
- クリップボード API が使えない場合のフォールバック（選択してコピー等）も用意
- 初学者向けに、イベント (e) から latlng を取る流れを説明

D.7 P04 (★ 2) Polyline/Polygon と hover ハイライト

到達目標

- 線（Polyline）・面（Polygon）を座標配列で描ける
- hover で一時ハイライトができる
- fitBounds で見切れなく表示できる

注意

Polygon は最後の点を最初に戻さなくても Leaflet が閉じるが、データによっては注意。

LLM プロンプト

Polyline と Polygon を描画する学習用サンプルを 1 ファイルで作って。

要件:

- Polyline を 2 本（見た目を変える：太さ/透明度など）
- Polygon を 1 つ（塗り/枠線）

- 全体が入るように fitBounds
- hover でハイライト、mouseout で元に戻す（線と面両方）
- 初学者向けに、Polyline と Polygon の違いを短く説明

D.8 P05 (★ 2) レイヤ切替 (Base/Overlay) +スケール+ズーム ON/OFF

到達目標

- 「ベース地図」と「重ねるレイヤ」の考え方分かる
- Layer Control を使える
- スクロールズームを ON/OFF できる

注意

ベースレイヤは通常 1つだけ表示、オーバーレイは複数 ON 可。

LLM プロンプト

ベースレイヤ 2つ+オーバーレイ 2つ（例: マーカー群、ポリゴン）を作り、Layer Control で切り替えできる例を作って。

要件:

- scale control 表示
- スクロールホイールズームは初期 OFF、ボタンで ON/OFF 切替
- いま表示中のレイヤ名を画面に出す（簡易で OK）
- 初学者向けに base/overlay の違いを説明

D.9 P06 (★ 3) クリックで選択状態（強調・解除）+サイドパネル

到達目標

- 「選択状態」を変数で管理できる
- hover (仮) と click (固定) を分けて実装できる
- 地図外 UI (サイドパネル) に properties を表示できる

注意

選択解除や「前の選択を戻す」処理が必要（状態管理の初歩）。

LLM プロンプト

GeoJSON（小さめで OK、コード内に埋め込み）を表示し、

クリックで「選択中スタイル」、もう一度クリックで解除する仕組みを作って。

要件:

- hover: 一時ハイライト
- click: 選択固定（他は薄く）
- 選択中 feature の properties を右側サイドパネルに表示（CSS 込み）
- 初学者向けに「状態管理（selectedFeature）」の考え方を説明

D.10 P07（★ 3）複数選択（Shift+ クリック）&選択リスト&GeoJSON 出力

到達目標

- 複数選択の状態管理（配列/Set）を学べる
- 選択一覧 UI を作れる
- 選択結果を FeatureCollection として出力できる

注意

Shift キー検出（イベントの `originalEvent.shiftKey` など）を理解する必要あり。

LLM プロンプト

地物（ポリゴン or マーカー）を Shift+ クリックで複数選択できる UI を作って。

要件:

- 選択中はスタイル変更
- 選択一覧（name/id）を画面に表示
- 「選択クリア」ボタン
- 選択を GeoJSON FeatureCollection として JSON 出力できる
- 初学者向けに「単一選択→複数選択に拡張する考え方」を説明

D.11 P08（★ 3）ポップアップ内ボタン→外部処理（安全な書き方）

到達目標

- popup 内のボタンをクリックして外部 UI を更新できる
- onclick 直書きを避けたイベント設計が分かる
- XSS の超基本（危険な文字列をそのまま HTML に入れない）が分かる

注意

popup が開くたびにイベントが多重登録されないように注意。

LLM プロンプト

ポップアップ内に「詳細を見る」ボタンを置き、クリックでサイドパネルに詳細を出す実装例を作つて。

要件:

- popup 内 HTML に直接 onclick を書かず、安全にイベントを紐付ける
- XSS 対策として properties 文字列を安全に扱う方針を説明（初心者向け）
- 多重登録を避ける注意点も書く

D.12 P09 (★ 3) 右クリックのコンテキストメニュー

到達目標

- 右クリックでメニューを出す UI を作れる
- 「追加・コピー・移動」など操作を地図に統合できる
- メニュー外クリックで閉じる等、基本の UI 作法が分かる

注意

ブラウザ既定メニューを抑制する (contextmenu イベント)。

LLM プロンプト

地図の右クリックで簡易コンテキストメニューを表示する例を作つて。

項目:

- ここにマーカー追加
- 座標コピー
- この地点に flyTo (ズーム +1)

要件:

- メニュー外クリックで閉じる
- モバイルでは長押し代替案も提示
- 初学者向けに、contextmenu イベントの意味を説明

D.13 P10 (★ 3) 現在地ボタン (Geolocation) + 誤差円

到達目標

- getCurrentPosition と watchPosition の違いが分かる
- 許可拒否/タイムアウトの扱いができる

- 現在地レイヤを「更新」して増殖させない実装ができる

注意

https://localhost 以外だと現在地が取れないことがある。

LLM プロンプト

「現在地へ移動」ボタンを追加し、現在地マーカー+精度 (accuracy) 円を表示して。

要件:

- 許可拒否/タイムアウト時のメッセージ
- 既存の現在地表示があれば更新（増殖させない）
- 現在地追従モード ON/OFF (ON の間だけ watchPosition)
- 初学者向けに、精度円 (accuracy) の意味を説明

D.14 P11 (★ 3) fetch で GeoJSON 読み込み（ローディング & エラー付き）

到達目標

- 外部ファイルを非同期で読み込んで描画できる
- ローディング表示／失敗表示が作れる
- 点・線・面で描画方法を変えられる

注意

file:///直開きは fetch が失敗しやすい → ローカルサーバ推奨。

LLM プロンプト

GeoJSON を fetch して描画する 1 ファイル例を作って。

要件:

- データ URL: <geojson_url> を自分の URL に置換（まずは埋め込みデータで動作確認→ 次に fetch）
- 読み込み中スピナー表示→完了で消える
- 失敗時のエラー表示（画面+ console）
- Point は circleMarker、Polygon は塗り、Line は線
- properties.name をポップアップに表示
- 初学者向けに「なぜローカルサーバが必要な場合があるか (CORS)」を説明

D.15 P12 (★3) CSV → マーカー (バリデーション込み)

到達目標

- CSV をパースして地図に出せる (データ加工の入口)
- 不正データを弾く、件数を表示する等の頑健性が身につく
- 値に応じた色分けができる

注意

緯度経度が文字列になっていることが多い → Number 変換とチェック。

LLM プロンプト

CSV (lat,lng,name,value) を読み込み、マーカー化する例を作って (外部ライブラリなし)。

要件:

- CSV は (1) 文字列埋め込み と (2)fetch の 2 パターン
- 不正データ (null/空/文字列 latlng) を弾き、弾いた件数も表示
- value で色分け (簡易の閾値で OK)
- 初学者向けに「バリデーションを入れる理由」を説明

D.16 P13 (★4) コロプレス+凡例+情報ボックス (テーマ地図の王道)

到達目標

- 段階色 (クラス分け) と凡例の仕組みが分かる
- 欠損値を別扱いできる
- hover 表示 + click 選択固定の定番構成を作れる

注意

値の分布に合わない閾値だと見づらい (最初は固定でも OK)。

LLM プロンプト

GeoJSON のポリゴンを properties.value でコロプレス表示し、凡例を自動生成して。

要件:

- hover で左下の情報ボックスに name/value を表示
- click で選択固定 (枠線太く)
- 欠損値は別色 & 凡例にも欠損枠を入れる
- 閾値 (クラス分け) をコード上で分かりやすく定義

- 初学者向けに「凡例がなぜ必要か」「段階色の作り方」を説明

D.17 P14 (★ 4) 地図+表の連動（双方）

到達目標

- 地図と DOM（表）を同期させる設計ができる
- 行クリック→地図移動、地図クリック→行ハイライトが作れる
- 小さな UI 操作（ソート等）を組み込める

注意

「どれが選択中か」を ID で一意に管理するのがコツ。

LLM プロンプト

地図上の点（配列データ）と、下の HTML テーブルを連動させる例を作って。

要件:

- 表の行クリック→地図が flyTo →該当マーカーを強調→popup 表示
- マーカークリック→表の該当行をハイライト & スクロールして見える位置へ
- ソート（name/value どちらか 1 つ）も簡易で付ける
- 初学者向けに「ID で紐付ける設計」の説明を入れる

D.18 P15 (★ 4) MarkerCluster（大量点の定番）

到達目標

- 大量点で「そのままマーカーを置くと重い」理由が分かる
- クラスタリングを導入して改善できる
- クラスタに「件数・内訳」を出せる

注意

プラグインは Leaflet 本体とは別（CSS/JS 両方が必要なことが多い）。

LLM プロンプト

MarkerCluster プラグインを CDN で導入し、大量点（例: 2000 点をランダム生成でも可）をクラスタ表示して。

要件:

- クラスタクリックでズーム
- クラスタのポップアップに「件数」「カテゴリ別内訳」を表示（カテゴリはランダムで OK）

- プラグインなしの代替 (Canvas/circleMarker 等で軽くする案) も併記
- 初学者向けに「プラグインとは何か」「導入で必要なもの (CSS/JS)」を説明

D.19 P16 (★ 4) Leaflet.draw (作図→編集→GeoJSON 出力)

到達目標

- ユーザーが地図上で図形を作れるようになる
- 作った図形を GeoJSON として保存できる
- 作成/編集/削除イベントを理解できる

注意

「描いたもの」を入れる専用レイヤ (FeatureGroup) が必要になるのが定番。

LLM プロンプト

Leaflet.draw を使って、点/線/面を描けて編集・削除できる例を作って。

要件:

- 描画結果を常に GeoJSON にしてテキストエリアへ出力
- 既存の GeoJSON を読み込んで編集開始できる (初期データを埋め込みで OK)
- created/edited/deleted イベントの違いを初学者向けに説明
- プラグイン導入 (CSS/JS) 手順も丁寧に

D.20 P17 (★ 5) 表示範囲内だけ取得 (debounce + Abort + 差分更新)

到達目標

- getBounds() を使った「範囲内データ取得」の考え方方が分かる
- debounce で過剰アクセスを防げる
- Abort で古い通信を止め、差分更新で地図を保つ実装ができる

注意

「同じ ID の点は更新、消えた点は削除」が肝 (データ構造が必要)。

LLM プロンプト

map.getBounds() を使って「表示範囲内だけ」API から取得する構成の例を作って (API はダミーで OK)。

要件:

- moveend/zoomend で debounce して取得
- 前の fetch は AbortController で中断
- 返ってきた点データを差分更新（追加/更新/削除）
- レート制限やサーバ負荷の注意点も書く
- 初学者向けに「なぜ debounce/Abort が必要か」を例で説明

D.21 P18 (★ 5) パフォーマンス比較の実験台（Marker vs CircleMarker vs Canvas）

到達目標

- ・「どの描画方式が重いか」を実測して理解できる
- ・DevTools での見方（ざっくりで OK）を学べる
- ・LOD（ズームで間引く）など実務の最適化を入れる

注意

最適化は「感覚」ではなく「測って決める」のが重要。

LLM プロンプト

Leaflet のパフォーマンス最適化を学ぶための「比較実験ページ」を 1 ファイルで作って。

要件:

- 同じ点データを (A)Marker (B)circleMarker (C)Canvas renderer で切替比較
- FPS 目安・操作感の評価方法（Chrome DevTools の見方）も説明
- ズームレベルで表示を間引く（LOD）実装も入れる
- どれを採用すべきか意思決定の指針をまとめる（初学者向けに）

D.22 P19 (★ 4) Vite + TypeScript で Leaflet（教材配布・実務の入口）

到達目標

- ・Vite 環境で Leaflet を動かせる
- ・TS で GeoJSON と properties に型を付けられる
- ・marker-icon のパス問題など定番の罠を回避できる

注意

「CSS の import」と「アイコンの参照」が初回につまずきやすい。

LLM プロンプト

Vite + TypeScript で Leaflet を動かす手順と最小コードを示して。

要件:

- Leaflet CSS 取り込み
- marker-icon のパス問題への対処（ありがちな罠）
- GeoJSON Feature の型付け例（properties に独自型を付ける）
- フォルダ構成案（map 初期化、layer 管理、データ取得を分離）
- 初学者向けに「なぜ分割するのか（保守性）」も説明

D.23 P20 (★3) デバッグ&レビュー（詰まつたらこれ）

到達目標

- 症状→原因候補→切り分け→修正の順で考えられる
- Leaflet 特有のトラブル（CSS 高さ、初期化順、CORS 等）に強くなる
- 再発防止チェックリストを持てる

注意

「地図が出ない」は 9 割が CSS か読み込み順（まずそこから）。

デバッグのコツ

貼るコードは最小再現（最短コード）に削ると、LLM も人間もデバッグが速くなります。機密情報は必ず伏せてください。

LLM プロンプト

次の Leaflet コードが <症状> で動きません。コンソールは <ログ> です。

初学者向けに、原因候補を「優先度順」に挙げ、切り分け手順→修正版コード→再発防止チェックリストまで出して。

特に以下を必ず点検:

- CSS 高さ/表示領域（html/body/#map）
 - 初期化順（DOM ready）
 - タイル URL/attribution/CORS
 - イベント多重登録やレイヤ増殖
- ここからコード—
- <ここに最小再現コードを貼る>
- ここまで—

D.24 おすすめの進め方

目的	推奨プロンプト
最短ルート（授業 1～2 回向け）	P01 → P02 → P03 → P05 → P06 → P11 → P13 → P20
実務寄り（ダッシュボード/研究）	上に加えて P14 → P15 → P17 → P18
作図や調査支援を作る	P03 → P09 → P10 → P16

表 D.2 目的別おすすめの進め方